FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Betting Experience Based On User Behaviour

João Pedro Fernandes Falcão Araújo

DISSERTAÇÃO

Orientador: Luís Filipe Teixeira (PhD)

Co-orientador: Filipa Fonseca Santos (MSc)

29 de Julho de 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"A Betting Experience Based On User Behaviour"

foi aprovada em provas realizadas em 20-07-2015

o júri

Presidente Professor Doutor Pedro Luís Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Pedro Manuel Henriques da Cunha Abreu
Professor Auxiliar Convidado do Departamento de Engenharia Informática da Universidade de Coimbra

Professor Doutor Luís Filipe Pinto de Almeida Teixeira
Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - João Pedro Fernandes Falcão Araújo

Faculdade de Engenharia da Universidade do Porto

Resumo

Atualmente, devido ao contínuo e exponencial crescimento da informação disponível na Internet, as pesquisas têm-se tornado mais complexas devido à necessidade de filtrar informação de forma a obter a mais desejada e credível. Este excesso de informação apresentado ao utilizador pode acontecer numa simples pesquisa, numa compra *online*, na escolha de um filme para ver, ou até na escolha de um mercado para apostar num site de apostas desportivas. Como consequência existe uma grande perda de informação desejada, uma vez que se torna bastante difícil para um utilizador filtrá-la.

Devido a este problema, foram criados os sistemas de recomendação, que têm como principal objetivo conseguir descartar a informação que o utilizador não deseja e, automaticamente, dar maior destaque àquela que apresenta um maior interesse para este. Estes sistemas são cada vez mais utilizados e extremamente úteis, proporcionando ao utilizador uma melhor e mais agradável experiência de utilização.

Esta dissertação começa por analisar o problema, traçar os objetivos do projeto, estudar os diferentes tipos de Sistemas de Recomendação e as categorias que integram os algoritmos de Aprendizagem Automática. A partir dessa análise, foi possível estudar quais os algoritmos e as tecnologias que melhor se enquadram no *website mobile* de *Exchange* da *Betfair*, realizando uma comparação entre elas.

Foi desenvolvida a ferramenta *Jarvis*, que surge para solucionar este problema de filtragem de informação no *Exchange Mobile Site*, onde foi recolhida informação sobre cada utilizador e, posteriormente, personalizado o aspeto do *website*, dando maior ênfase aos mercados de aposta mais interessantes ao utilizador. Este processo recorre a dados de utilizadores ou a perfis semelhantes para, automaticamente, personalizar a informação que irá ser apresentada.

Posteriormente, foi ainda criado um conjunto de testes que verificam a funcionalidade das recomendações *Jarvis*, a eficiência da ferramenta desenvolvida e dos módulos alterados. Estes testes encontram-se detalhados em termos de aumento no tempo de resposta da aplicação, quer a nível do carregamento dos dados dos módulos como a nível de serviços que comunicam com a base de dados.

A partir desta dissertação, pode concluir-se que é possível incluir num site de apostas desportivas um sistema de recomendação, sem que a performance deste seja significativamente afetada. De destacar ainda que o *Jarvis* pode incluir um maior número de categorias a recolher de forma a aumentar o espectro das suas recomendações.

Abstract

Nowadays and due to the continuous and exponential growth of the available information on the Internet, searching became more complex due to the need of filtering information to obtain the most desired and credible one. This overload of information can be represented in a simple search, in an online store purchase, when choosing a movie or a market to bet in a sports betting website. As a consequence, there is a huge lost of desired information, because it became very hard for a person to filter this data.

To solve this issue, the recommendation systems were created, whose main goal is to filter information that the user doesn't want and, automatically, give more emphasis to the information more suitable to his interests. The use of these systems is increasing and they are becoming essential to the proper use of services. These tools are extremely useful providing a much better user experience.

This master's dissertation starts by analysing the problem, outlining the main goals of the project, studying the types of Recommendation Systems and the categories of Machine Learning algorithms. From this analysis, it was possible to study which algorithms and technologies fit better on the Betfair's Exchange Mobile Site, by doing a comparison between them.

The "Jarvis"solution was developed to solve this problem of information filtering in the Exchange Mobile Site, by collecting information about each user which resulted in a customization of the information displayed, giving more emphasis to the markets more interesting to the user. This process requires the use of previously collected data to, automatically, customize the information for each user.

It was also created a set of tests that verify the "Jarvis"Recommendations functionalities, the efficiency of the developed solution and the updated modules. This tests were detailed in terms of the growth of the application response time, whether in terms of load of data or in services that communicate with the database.

Finally, it was concluded that it's possible to include a recommendation system in a sports betting website, without degrading a lot the performance. It's also important to highlight that "Jarvis"can collect more categories and therefore grow the recommendation spectrum.

Agradecimentos

Numa dissertação é fundamental o trabalho individual, no entanto existem pessoas que de uma forma ou outra, foram importantes no desenvolvimento desta. A essas pessoas que deram essa contribuição quero agora agradecer.

Em primeiro lugar, gostaria de agradecer aos meus orientadores Prof. Luís Filipe Teixeira e Engenheira Filipa Santos por todo o tempo dispensado, por todos conselhos e pela ajuda fundamental à realização desta dissertação.

À Blip, ao EMS e em especial à *Bushido*, por todo o apoio prestado e por toda a disponibilidade que sempre demonstraram para me ajudar. Um agradecimento especial ao Joaquim Rocha que sempre se mostrou disponível para me ajudar e que foi muito importante na forma como me ajudou a conduzir o desenvolvimento do projeto.

Aos meus pais e irmão, por me terem apoiado, por toda a compreensão e por me terem motivado a realizar a dissertação.

Finalmente à minha namorada, por me ter ouvido vezes sem fim, por me ter motivado, compreendido e ajudado a ir sempre mais além, dando sempre o meu melhor.

A todas estas pessoas, mais uma vez, os meus sinceros agradecimentos.

Porto, Junho de 2015

"Any fool can know.
The point is to understand."

Albert Einstein

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

EMS	Exchange Mobile Site
IA	Inteligência Artificial
SM	Simple Matching Coefficient
JC	Jaccard
RR	Russel and Rao
CF	Cluster Feature
k-NN	k-Nearest Neighbors
API	Application Programming Interface

Capítulo 1

Introdução

Antigamente, a procura de informação era uma tarefa bastante morosa e que poderia ser infrutífera. No entanto, nos dias que correm, esse problema já não ocorre tão frequentemente mas, pelo contrário, tornou-se bastante complicado filtrar a informação relevante e pertinente para cada ocasião. Por tudo isto, pela quantidade de informação que chega todos os dias à Internet, é fundamental existir uma seleção do que é apresentado e personalizar a informação a disponibilizar para cada utilizador [3].

Devido aos problemas apresentados anteriormente, a personalização de conteúdos tem vindo a destacar-se. É expectável que, cada vez mais, a informação seja customizada a cada utilizador/cliente, quer seja no mundo tecnológico ou até na área dos serviços, tornando a personalização de conteúdos uma das principais chaves para o sucesso futuro dos motores de pesquisa e *websites*.

Nesta dissertação será apresentada uma proposta de solução, o *Jarvis*, para um sistema de personalização de um site de apostas desportivas, a *Betfair* [4]. Este capítulo contextualiza o tema, as motivações que levaram este projeto a ser desenvolvido, define o problema, os objetivos, os resultados esperados, o trabalho relacionado e, por fim, a estrutura que irá ser seguida no relatório.

1.1 Enquadramento e Motivação

A quantidade de informação que pode ser encontrada na Internet está sempre a aumentar, e com ela muitos problemas surgem no que toca à filtragem. Devido a este excesso de informação, o tempo gasto a filtrar e encontrar a informação que realmente é desejada e credível tem tendência a aumentar.

Contudo, alguns *websites* já contêm algumas ferramentas de costumização e por isso já conseguem personalizar e/ou recomendar os seus produtos ou pesquisas às reais necessidades do utilizador. Nesse sentido, existe uma necessidade assinalável dessas ferramentas serem utilizadas também nas páginas de apostas desportivas, que ainda não correspondem às necessidades dos seus utilizadores, uma vez que existe um número muito grande de mercados passíveis de aposta e as possibilidades de aposta dentro de um simples jogo são inúmeras. Isto implica que exista um número consideravelmente grande de possibilidades e possíveis alvos para um apostador e como

tal, é necessária uma filtragem daquilo que é apresentado e que será de maior utilidade a cada utilizador.

Com a personalização dentro dos motores de apostas, serão obtidas inúmeras vantagens e em vários aspetos, isto beneficiará tanto os utilizadores como a própria casa de apostas. Para começar, com esta ferramenta funcional, os apostadores sairiam bastante beneficiados, porque para além de obterem uma muito melhor experiência de aposta, conseguiam chegar aos mercados e jogos desejados mais rapidamente e isso poderá representar lucros para estes. Para as casas de apostas isto também é bastante útil pois é traduzido numa navegação e utilização mais agradável e em utilizadores mais satisfeitos, o que normalmente implicará mais utilizadores e maior receita.

Para esta personalização a cada utilizador, é importante que para cada tarefa, sendo ela uma pesquisa ou uma aposta, seja anteriormente recolhida informação relevante acerca do utilizador e que com essa informação, as páginas *web* possam ser personalizadas com sistemas de recomendação que correspondam aos interesses dos utilizadores [5].

Esta solução será incluída na página *mobile* de *Exchange* da *Betfair* (EMS), que é um *site* para dispositivos móveis onde é possível realizar apostas desportiva.

1.2 Objetivos

Esta dissertação tem como objetivo criar uma ferramenta de aprendizagem automática (*Machine Learning*) ligada às apostas desportivas, que fornece informação personalizada e uma experiência de navegação focada no utilizador.

Para a realização deste projeto, o trabalho irá ser distribuído por etapas. Inicialmente, será elaborado um relatório acerca do estado da arte de sistemas de recomendação em apostas desportivas. Posteriormente, será recolhida informação acerca dos utilizadores. Será desenvolvida a ferramenta *Jarvis* que implementará um sistema de recomendação e finalmente será adaptada a aplicação baseada na informação recolhida e realizados testes experimentais para avaliar a eficiência desta personalização.

1.3 Descrição do Problema

Apesar de ser um problema essencial nos dias de hoje e de já existirem muitos projetos que utilizam algoritmos de *Machine Learning* que forneçam recomendações ao utilizadores, a verdade é que nas apostas desportivas existem muito poucos projetos que utilizem estas técnicas. No caso do EMS, já existe personalização quanto aos conteúdos apresentados, no entanto apenas é costumizada a informação mediante o país, não se adaptando ao perfil de cada utilizador.

Neste projeto, será muito importante recolher informação sempre que o utilizador efetua *login*, registando o histórico de apostas e páginas visualizadas por este, para posteriormente conseguir obter quais as suas equipas, competições e até desportos favoritos.

Depois da informação de todos os clientes ser recolhida é essencial customizar a informação para cada utilizador. Contudo, existem particularidades a ter em conta, como o caso de adaptar

a página *web* a utilizadores com pouco histórico de apostas e até com um pequeno histórico de navegação. Aqui, será fundamental criar grupos de utilizadores com gostos semelhantes e assim, conseguir colmatar a falta de informação acerca do utilizador através de informações de outros utilizadores que se enquadrem no mesmo grupo. No entanto, para não obter resultados inesperados ou inadequados na personalização de conteúdos, é necessário ter muitos cuidados na recolha de informação e posterior aglomeração dos grupos.

Neste projeto, é necessário conseguir adaptar conteúdos a cada utilizador garantindo que a experiência de utilização do mesmo não saia prejudicada e ao mesmo tempo, conseguir que o utilizador não se sinta obrigado a apostar naqueles desportos/competições que costuma apostar, sendo que para isso, as suas preferências sejam complementadas pelas de utilizadores semelhantes. Outra questão essencial do trabalho é que, terá de ser tomado em conta o facto deste algoritmo ser implementado num *site mobile* por isso, o impacto que este pode ter no desempenho atual deverá ser muito reduzido.

1.4 Resultados

No que toca a resultados, foi desenvolvida uma solução que personaliza o EMS da *Betfair* consoante o perfil de cada utilizador, sendo que para isso são recolhidas informações acerca destes, nomeadamente os históricos de visualização e de apostas.

Estas alterações podem ser ativadas ou desativadas pelo utilizador a qualquer momento na aplicação, recorrendo a um botão desenvolvido que troca a experiência de utilização e foi integrado um algoritmo de *Machine Learning* que corresponde às necessidades do produto.

Com este projeto, foi possível criar uma melhor experiência no EMS, alcançando uma pesquisa mais eficiente dos desportos ou mercados favoritos de cada utilizador.

1.5 Estrutura da Dissertação

Este documento está dividido em seis capítulos: Introdução, Sistemas de Recomendação, *Machine Learning*, Especificação e Implementação da Ferramenta, Resultados Experimentais e Conclusões.

O primeiro capítulo apresenta qual é o tema deste projeto, o enquadramento e a motivação, os principais objetivos, a descrição do problema e, por fim, os resultados do mesmo.

No capítulo 2, serão apresentados os principais algoritmos de Sistemas de Recomendação, quais as suas características, a razão para serem utilizados e os sistemas de recomendação nas Apostas Desportivas. No final do capítulo serão apresentadas algumas conclusões acerca das categorias mais adequadas para a elaboração deste projeto.

De seguida, será descrito o capítulo 3 referente à Aprendizagem Automática para Sistemas de Recomendação, onde será explicado o que é *Machine Learning*, quais os tipos de algoritmos e quais os que se destacam no contexto pretendido. No final serão apresentadas as técnicas utilizadas

no âmbito das apostas desportivas e as conclusões que se retiram deste estudo para o desenvolvimento da solução.

O capítulo 4 corresponde ao tópico Especificação e Implementação da Ferramenta, onde serão apresentados os requisitos e objetivos, a arquitetura, as tecnologias utilizadas, todo o trabalho desenvolvido para a elaboração da solução, as optimizações realizadas para melhorar a performance e um resumo, que apresenta algumas conclusões que se podem retirar deste capítulo.

No capítulo 5 são apresentados os resultados experimentais realizados à solução desenvolvida, quer testes de funcionalidade como testes de desempenho, por fim serão resumidas as conclusões a retirar dos testes desenvolvidos.

Por fim, o capítulo 6 refere-se às conclusões da dissertação acerca do estudo, desenvolvimento e análise da ferramenta desenvolvida.

Capítulo 2

Sistemas de Recomendação

Neste capítulo será explicado o que são Sistemas de Recomendação, quais são as suas principais vantagens, tanto para utilizadores como empresas, as técnicas inerentes à utilização destes sistemas e as categorias que os classificam.

No fim deste capítulo, será apresentado um resumo das ilações que se podem retirar acerca deste tópico, indicando quais são os métodos mais adequados ao projeto.

2.1 Âmbito

Com o crescimento exponencial de utilização da Internet, com o contínuo crescimento da informação e também com o crescimento do comércio *online* foi necessário o desenvolvimento de Sistemas de Recomendação. Estes sistemas são responsáveis pela filtragem personalizada de informação, usada para identificar os conjuntos de produtos que serão do interesse de cada utilizador [6]. Os sistemas de recomendação são utilizados tanto para prever se um determinado utilizador irá gostar de um certo produto, como identificar um conjunto de n produtos que serão do interesse de determinado utilizador.

Atualmente, os sistemas de recomendação são ferramentas extremamente úteis e utilizadas nos mais variados tipos de mercado. Dois dos exemplos mais conhecidos destes sistemas são os utilizados nas páginas da *Amazon* [7] e *Netflix* [8], onde em ambos os casos, são sugeridos produtos/filmes semelhantes aos que o utilizador já comprou ou viu.

Os sistemas de recomendação podem ser classificados em duas principais categorias: *Content-based systems* e *Collaborative filtering systems*. A primeira examina as propriedades dos produtos recomendados, ou seja: se um utilizador compra frequentemente produtos de um conjunto, então o sistema irá recomendar outros produtos que o utilizador não comprou, dentro desse mesmo conjunto. Na segunda categoria, os produtos recomendados são baseados em medições de semelhança entre produtos e/ou utilizadores, isto é, os produtos sugeridos são aqueles que são preferidos por utilizadores semelhantes [1].

As diferenças entre *Content-based systems* e *Collaborative filtering systems* podem ser expressas recorrendo a dois sistemas de recomendação de música bastante populares, Pandora [9] e Lastfm [10].

O Pandora utiliza um *Content-based system*, pois recorre às propriedades das músicas ou artistas (utilizando inúmeros atributos) de forma a fornecer uma rádio de músicas com propriedades semelhantes. O *feedback* recebido é utilizado para redefinir os resultados das rádios, alterando os pesos de cada atributo.

No caso do Lastfm, este recorre a um *Collaborative filtering system*, uma vez que se baseia nas bandas e músicas que o utilizador costuma ouvir regularmente, comparando-as com os hábitos de outros ouvintes semelhantes, tendo como resultado a criação de uma estação de rádio.

De seguida, serão apresentadas estas duas categorias, explicando as principais características de cada uma delas.

2.2 *Content-based systems*

Nas aplicações que utilizam sistemas *Content-based*, a informação é recolhida consoante os hábitos de cada utilizador. Por exemplo, numa aplicação de recomendação de filmes, os filmes sugeridos são baseados nas avaliações realizadas pelo utilizador que podem ser referentes a atores, realizadores e temáticas abordadas. A partir das avaliações efetuadas, apenas os filmes que têm grande semelhança com aqueles que obtiveram boa classificação é que são sugeridos.

Neste tipo de sistemas, é necessário construir para cada produto um perfil, que é uma representação das características mais importantes desse mesmo produto. Em situações mais simples, o perfil consiste nas características deste que são facilmente descobertas. Por exemplo, no caso de um filme, as características que poderão ser importantes para um sistema de recomendação serão: os atores envolvidos no filme, o realizador, o ano em que o filme foi feito, o género de filme, entre outros. Para criar o perfil do produto, usualmente são utilizadas variáveis binárias para representar cada característica, no entanto algumas delas não são representáveis por valores binários, por exemplo a média da avaliação de cada filme.

Depois de criar o perfil do produto, é necessário criar o perfil do utilizador, onde são resumidas as preferências de cada utilizador, baseando-se na sua utilização. Aqui, é necessário criar uma tabela que represente o perfil do utilizador, onde os valores podem ser representados de várias formas, mas devem conter toda a informação acerca das preferências deste. Por exemplo, se o ator *Christian Bale* participa em 30% dos filmes que o utilizador gosta, então o ator será representado na tabela do utilizador com 0,3.

Com os perfis dos produtos e dos utilizadores já representados, é necessário criar uma forma de sugerir ao utilizador os produtos mais relacionados com ele, para isso é necessário estimar o grau de proximidade entre o utilizador e os produtos. Dependendo das características que são utilizadas para a representação dos perfis, a forma de calcular a semelhança entre os produtos e utilizador varia. Por exemplo, caso não seja representada a avaliação que o utilizador dá ao filme,

contendo apenas se gosta ou não do filme e quais os atores envolvidos, então a sugestão dos filmes irá ser feita apenas tendo em conta os filmes que contenham os atores que o utilizador gostou.

Uma abordagem completamente diferente é tentar resolver o problema como um de *Machine Learning*. Neste caso, poderia ser utilizada uma árvore de decisão para ajudar a sugerir os produtos.

Tal como foi referido no capítulo 3.2.1, as folhas numa árvore de decisão são utilizadas para apresentar as decisões. Neste exemplo, cada folha será utilizada para decidir se o utilizador gosta ou não de determinado produto. A árvore começa com uma raiz que representa um grupo mais genérico, posteriormente, sairão dois ou mais filhos, onde cada grupo será subdividido em mais pequenos até chegar às nós finais (folhas), que serão "gosta"ou "não gosta". A figura 2.1 ilustra um pequeno exemplo do uso de uma árvore de decisão no contexto de recomendação de filmes.

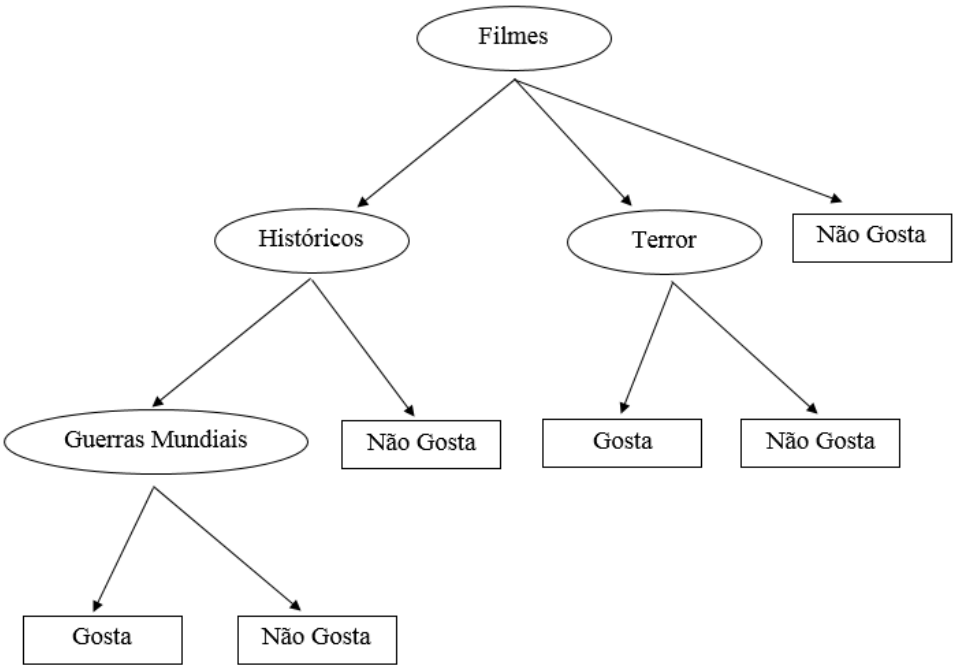

Figura 2.1: Exemplo utilizando uma Árvore de Decisão

2.3 *Collaborative filtering systems*

Nos *Collaborative filtering systems*, ao contrário dos sistemas *content-based*, o sistema tem como objetivo prever a utilidade de determinados produtos para um utilizador em particular, baseando-se nos produtos avaliados anteriormente por outros utilizadores [11]. Formalmente, a utilidade $u(c,s)$ do produto s para o utilizador c é estimada tendo em conta as utilidades $u(c_j,s)$ atribuídas ao produto s pelo utilizador $c_j \in C$ que são similares ao utilizador c.

Dentro desta categoria de sistemas, existem dois tipos de recolha de dados: Implícita ou Explícita.

A recolha de dados implícita é realizada observando os produtos que o utilizador vê na loja *online*, analisando o número de visualizações de cada produto e registando cada produto comprado. Já a recolha de dados explícita é feita pedindo ao utilizador para avaliar os produtos, para criar uma lista de produtos favoritos, ou até produtos que gosta ou não, e por exemplo, pedindo ao utilizador para escolher entre dois produtos qual o que ele prefere.

Independentemente de qual a forma de recolher os dados, é necessário decidir qual a forma como irá ser medida a semelhança entre utilizadores e produtos.

Dentro desta categoria existem alguns desafios a serem resolvidos para ser possível criar um sistema de recomendação. Nomeadamente, é necessário ter em conta que na maior parte dos sistemas, mesmo os utilizadores mais ativos devem ter comprado bastante menos de 1% dos produtos dispostos para venda, por isso é difícil obter recomendações precisas. Para além disso, outra questão importante é o facto de a necessidade de processamento aumentar exponencialmente quer com o crescimento do número de utilizadores quer com o número de produtos [12].

Para além dos problemas já apresentados, é muito difícil calcular a semelhança entre utilizadores e produtos quando o sistema detém pouca informação sobre estes. Uma das formas de lidar com este problema é agrupar os utilizadores e/ou produtos, utilizando *clusters*. Para criar estes grupos podem ser os utilizados os métodos apresentados no capítulo 3.3.1. Contudo, é aconselhável utilizar um número pequeno de *clusters*, por exemplo, criar um número de grupos correspondente a metade do número de produtos existentes.

Dentro desta categoria existem vários métodos passíveis de serem utilizados, como por exemplo: *User–User Collaborative Filtering, Item–Item Collaborative Filtering, Probabilistic Methods* e *Hybrid Recommenders*.

2.3.1 *User–User Collaborative Filtering*

User–User Collaborative Filtering foi o primeiro método automático de *Collaborative Filtering*. Este método é bastante direto, procurando utilizadores cujas avaliações anteriores sejam similares ao utilizador desejado e utilizar essas avaliações para prever outros produtos que o utilizador irá gostar.

Para gerar as predições ou recomendações para o utilizador u, inicialmente é calculada a vizinhança $N \subseteq U$ dos vizinhos de u. Aqui o sistema combina as avaliações dos outros utilizadores em N para gerar as recomendações para o utilizador u do produto i. Normalmente é utilizada uma média das avaliações dos outros utilizadores para o produto i usando a semelhança como medida.

$$p_{u,i} = \bar{r_u} + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r_{u'}})}{\sum_{u' \in N} |s(u,u')|} \tag{2.1}$$

Uma decisão importante na implementação *user-user* é a escolha da função para medir a semelhança. Existem vários métodos que permitem realizar esta função, nomeadamente a *Pearson correlation, Jaccard coefficient, Spearman rank correlation* e *Cosine similarity*.

Pearson Correlation

Este método realiza uma correlação estatística entre avaliações de dois utilizadores, de forma a avaliar a sua proximidade. A correlação é medida utilizando a equação 2.2.

$$s(u,v) = \frac{\sum_{i \in I_u \bigcap I_v}(r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in I_u \bigcap I_v}(r_{u,i} - \bar{r_u})^2}\sqrt{\sum_{i \in I_u \bigcap I_v}(r_{v,i} - \bar{r_v})^2}} \tag{2.2}$$

Esta medida tem como problema calcular semelhança entre utilizadores com poucas avaliações em comum.

Jaccard Coefficient

O Jaccard Coefficient é um método que permite calcular a semelhança e diversidade entre dois conjuntos. Este é medido utilizando o tamanho da interseção dividido pelo tamanho da união de ambos os conjuntos.

$$s(u,v) = \frac{\mid u \cap v \mid}{\mid u \cup b \mid} \tag{2.3}$$

Spearman rank correlation

Este método classifica o produto com avaliação do utilizador mais alta com valor 1 e o produto avaliado com pior classificação com o valor mais alto. Produtos com a mesma avaliação são classificados com o valor médio da sua posição. Posteriormente, os cálculos utilizados são os apresentados na equação 2.2, no entanto em vez de avaliações são utilizadas as classificações atribuídas.

Cosine Similarity

Este modelo tem uma abordagem bastante distinta das anteriores pois este tem uma abordagem baseada na álgebra linear em vez de estatística. Aqui, os utilizadores são representados num vetor bidimensional e a proximidade entre estes é medida calculando a distância entre os dois vetores de avaliação.

$$s(u,v) = \frac{r_u . r_v}{\|r_u\|\|r_v\|} = \frac{\sum_i r_{u,i} r_{v,i}}{\sqrt{\sum_i r_{u,i}^2}\sqrt{\sum_i r_{u,i}^2}} \tag{2.4}$$

2.3.2 *Item–Item Collaborative Filtering*

Apesar do método *user-user collaborative filtering* ser efetivo, sofre de problemas de escalabilidade com o crescimento da informação sobre os utilizadores. Neste método, em vez de serem utilizadas as semelhanças entre utilizadores, são utilizadas semelhanças entre avaliações de produtos. Se dois produtos tendem a ter o mesmos utilizadores a gostarem ou não deles, então eles são similares e os utilizadores tendem a gostar da mesma forma de produtos semelhantes.

Com a existência de avaliações sobre os produtos, os valores de similaridade entre produtos podem ser utilizados para gerar predições utilizando a média, semelhante ao procedimento utilizado no *user-user collaborative filtering*. As recomendações são geradas escolhendo o produto com o valor mais alto de predição.

Depois de recolher um conjunto S de produtos similares a i, $p_{u,i}$ pode ser calculado.

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) t_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{2.5}$$

Para calcular a semelhança entre os produtos, existem vários métodos que podem ser utilizados, por exemplo, *Pearson correlation, Cosise Similarity* e *Conditional Probability*. Os dois primeiros métodos foram apresentados no tópicos 2.3.1 e 2.3.1 respetivamente.

O método ***Conditional Probability*** é utilizado para situações onde são utilizadas avaliações unárias. A função de semelhança é baseada em probabilidades condicionais. Neste método B é o histórico de compras de um utilizador e α é um parâmetro para equilibrar a ocorrência de produtos frequentes.

$$s(i,j) = \frac{Freq(i \bigwedge j)}{Freq(i)(Freq(j))^{\alpha}} \tag{2.6}$$

2.3.3 *Probabilistic Methods*

O método tem como objetivo calcular $P(i|u)$, a probabilidade do utilizador u comprar ou visualizar o produto i, ou $P(r_{u,i}|u)$, a distribuição da probabilidade da avaliação do utilizador u ao produto i. Baseado no histórico de compras do utilizador, este método estima a probabilidade de um utilizador comprar o produto a, sabendo que este comprou o produto b.

2.3.4 *Hybrid Recommenders*

Tal como o próprio nome indica, este método combina vários tipos de algoritmos num único sistema. Este tipo de abordagem tem grandes vantagens sobretudo porque utilizando mais que um algoritmo é possível contornar algumas dificuldades evidenciadas por determinados métodos. Por exemplo, o *item-item collaborative filtering* tem como problema a inexistência de produtos avaliados. Com este método, é possível utilizar as características dos produtos e juntamente com os produtos favoritos do utilizador, conseguir recomendar produtos adequados ao utilizador, contornando um problema existente [13].

Dentro da categoria de Sistemas de recomendação híbridos, existem várias abordagens.

- *Weighted recommenders* - utiliza as avaliações produzidas por vários métodos e gera uma lista de recomendações para o utilizador.
- *Switching recommenders* - troca entre diferentes algoritmos e utiliza aquele que tem o melhor resultado esperado para cada situação.
- *Mixed recommenders* - apresenta o resultado de vários métodos juntos.

- *Feature-combining recommenders* - utiliza várias fontes de informação como *input* e e utiliza um único algoritmo.

- *Cascading recommenders* - recorre a uma cadeia de algoritmos, onde a saída de um é a entrada do seguinte.

Com este método é possível combinar as vantagens de vários algoritmos de forma a obter as vantagens de cada um deles.

2.3.5 Exemplo

No exemplo presente na figura 2.2, apresentado por Rajaraman [1], serão demonstradas as avaliações de quatro utilizadores A, B, C, D a sete filmes HP1, HP2, HP3, TW, SW1, SW2, SW3.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Figura 2.2: Exemplo de Avaliação de Filmes [1]

Como pode ser observado, os utilizadores A e C avaliaram dois filmes em comum, no entanto, as avaliações são muito díspares. Por isso mesmo, é esperado que o resultado da medição da semelhança destes utilizadores os coloque em posições opostas.

Utilizando o método *Cosine Similarity* para medir a distância entre os utilizadores A-B e A-C, os resultados seriam os seguintes.

$$A-B = \frac{4\times 5}{\sqrt{4^2+5^2+1^2}\sqrt{5^2+5^2+4^2}} = 0.380$$

$$A-C = \frac{5\times 2+1\times 4}{\sqrt{4^2+5^2+1^2}\sqrt{2^2+4^2+5^2}} = 0.322$$

Quanto maior for o valor obtido no cálculo, implica um ângulo menor, ou seja, uma distância menor entre utilizadores. Com os valores obtidos nos exemplos, pode concluir-se que A está mais próximo de B do que C.

Para demonstrar um possível procedimento, foram criados *clusters* a partir dos filmes apresentados na figura 2.2. Para a criação dos *clusters*, foram aglomerados os filmes da saga do *Harry Potter* (HP) e da trilogia do *Star Wars* (SW), obtendo o resultado seguinte.

	HP	TW	SW
A	4	5	1
B	4.67		
C		2	4.5
D	3		3

Figura 2.3: Exemplo de Avaliação dos Utilizadores a *Clusters*

O valor apresentado como avaliação em cada um dos grupos, é dado pela média aritmética das avaliações do utilizador aos filmes que compõem cada grupo. Depois de terem sido criados estes grupos e terem sido medidas as distâncias entre eles, é possível agrupar recursivamente até obter o número desejado de grupos de produtos. Posteriormente é possível agrupar utilizadores, e tal como com os produtos, o processo é repetido até ser atingido o número de *clusters* pretendido.

Depois de terem sido criados os grupos desejados, é possível proceder às medições das distâncias entre os *clusters* e utilizá-los para a recomendação desejada.

2.4 Dificuldades e Limitações

Os Sistemas de Recomendação normalmente passam por quatro importantes dificuldades: *Cold Start*, Escalabilidade, *Sparsity* e Fraude.

Cold Start

A inclusão de novos produtos ou registo de novos utilizadores é um grande desafio aos sistemas de recomendação. Estes problema normalmente surge nos *Collaborative Filtering systems*, onde um produto não pode ser recomendado se nunca foi avaliado por nenhum utilizador. Este problema surge também no registo de um novo utilizador, pois este ainda não tem registadas as preferências e por isso é mais complicado encontrar utilizadores semelhantes.

Estes sistemas requerem uma grande quantidade de informação sobre o utilizador para poder fornecer recomendações mais precisas. Devido a isto, quando o utilizador não é frequente e/ou recente, o sistema tem dificuldades em recomendar produtos corretamente.

Escalabilidade

Devido ao facto destes sistemas estarem introduzidos num meio com milhões de produtos e utilizadores, é necessário um enorme poder de processamento para realizar recomendações, o que se torna ainda mais difícil com o número crescente de produtos e utilizadores a analisar.

Sparsity

Normalmente estes sistemas estão introduzidos num meio onde a percentagem de produtos avaliados pelo utilizador é muito baixa e mesmo o produto mais popular tem poucas avaliações, o

que dificulta a sugestão de produtos adequados ao utilizador. Muitos utilizadores não avaliam os produtos e por isso, ao utilizar um sistema de *Collaborative Filtering*, a probabilidade de encontrar um conjunto de utilizadores com avaliações semelhantes diminui.

Fraude

Como os sistemas de recomendação estão a ser cada vez mais adotados por lojas *online*, estes estão a ter um papel mais importante no que toca aos lucros dos vendedores. Este facto levou a que muitos tendam a avaliar os seus próprios produtos com altas avaliações e baixas nos produtos concorrentes, sendo que neste momento, este problema é muito estudado.

2.5 Sistemas de Recomendação nas Apostas Desportivas

Apesar dos Sistemas de Recomendação serem comuns nos dias de hoje, a verdade é não são conhecidos exemplos de utilização no âmbito das Apostas Desportivas. Nesta área, são muito utilizadas ferramentas de *Machine Learning*, no entanto são sobretudo relacionadas com cálculos de probabilidades de vitória de cada equipa e também cálculo de *odds* de cada um dos mercados.

2.6 Resumo

Atualmente, existe uma grande variedade de sistemas de recomendação utilizados, por isso mesmo, existe uma vasta gama de algoritmos. Contudo existe uma nova abordagem denominada de híbrida, pois utiliza tanto os *Collaborative filtering systems* como os *Content-based systems*, que, por vezes, consegue obter resultados mais eficientes.

Apesar de existirem várias técnicas e procedimentos a utilizar, são precisos muitos cuidados com o tratamento dos dados pois, caso isso não aconteça, podem levar a situações indesejáveis, ou seja, sugestões erradas.

Apesar dos métodos *Content-based* evitarem muitos dos problemas apresentados em 2.4, os sistemas *Collaborative Filtering* continuam a possuir vantagens em relação aos restantes. Isto porque este trabalha em domínios onde não existe muita informação acerca dos produtos ou são dados poucos objetos para processar [14].

Depois de terem sido estudadas as técnicas e analisadas quais as categorias que integram os sistemas de recomendação, é possível concluir que, no contexto desta dissertação, será mais indicado a utilização de um método de recolha de dados implícito, visto que não é esperado pedir aos utilizadores para especificar quais são as competições favoritas, nem pedir para avaliar se gosta ou não de determinados desportos ou equipas.

Capítulo 3

Aprendizagem Automática para Sistemas de Recomendação

Depois de no capítulo 2 terem sido apresentados os sistemas de recomendação, onde foram identificadas as principais categorias, neste capítulo será feita uma breve introdução sobre *Machine Learning* aplicado aos sistemas de recomendação, incidindo por isso sobretudo sobre os algoritmos que mais se adequam à solução a desenvolver e são mais indicados para esta aplicação. Depois serão resumidos alguns dos principais algoritmos, os seus prós e contras, e por fim concluir quais são os tipos de algoritmos mais proveitosos para este projeto.

3.1 Âmbito

Machine Learning é uma disciplina de estudo que trata uma enorme quantidade de dados, utilizando a experiência e conhecimento associado com Inteligência Artificial (AI) para realizar tarefas tais como identificação, reconhecimento, planeamento, controlo ou predição [15].

Para resolver qualquer problema utilizando *Machine Learning*, é necessário um algoritmo, isto é, uma sequência de instruções que são utilizadas para transformar um *input* do problema num *output* [16]. Existem inúmeros algoritmos de aprendizagem automática e em cada um dos casos pode haver mais do que um algoritmo adequado e útil. No entanto, a chave é utilizar aquele que consegue obter melhores resultados no compromisso entre eficiência e complexidade. Muitas vezes são escolhidos algoritmos com menor complexidade, necessitando de um menor número de instruções e até de memória para conseguir obter um resultado aceitável, não comprometendo a sua eficiência.

Os algoritmos de *Machine Learning* estão normalmente divididos nas seguintes três categorias: Aprendizagem Supervisionada, Aprendizagem não Supervisionada e Aprendizagem por Reforço.

Aprendizagem Supervisionada

Neste tipo de aprendizagem, é utilizado um professor, isto é, uma entidade superior que fornece ao computador exemplos de entradas e saídas desejadas. O computador irá aprender as regras

a utilizar, a partir dos exemplos fornecidos, para transformar os dados recebidos em *outputs* do sistema.

Aprendizagem não Supervisionada

Neste caso, não existe nenhum tipo de supervisor ou professor. O objetivo da aprendizagem não supervisionada é encontrar padrões nos valores de entrada sem receber qualquer tipo de informação de ordem superior. Aqui, a máquina tem de descobrir a estrutura das entradas e autonomamente transformá-las na saída do sistema.

Aprendizagem por Reforço

Neste tipo de algoritmos, não existe nenhum tipo de padrão de transformação de *inputs* em *outputs*. Aqui, o sistema vai adaptando-se ao longo do tempo, aprendendo com os próprios erros de forma a conseguir atingir os objetivos, utilizando o método tentativa erro.

Das três categorias de *Machine Learning* apresentadas, a que menos se enquadra no âmbito desta dissertação é a Aprendizagem por Reforço, visto que não é esperado adaptar o resultado consoante o valor obtido, adaptando o seu *output* utilizando a tentativa erro. Apesar de não existir uma categoria mais adequada, tendencialmente será preferível utilizar uma Aprendizagem não Supervisionada pois não existem padrões de transformação de *input* em *output*. No entanto algumas técnicas de Aprendizagem Supervisionada poderão ser úteis para os sistemas de recomendação. Por tudo isto, de seguida serão apresentadas as principais categorias dentro destes dois tipos de aprendizagem e, para além disso, serão apresentados alguns exemplos acerca das abordagens de cada tipo de algoritmo.

3.2 Aprendizagem Supervisionada

Na Aprendizagem Supervisionada existe uma grande variedade de métodos de implementação. Contudo, aqueles que são mais comuns são: as Árvores de Decisão, as Redes Neuronais Artificiais, o *Naive Bayes* e o *K-Nearest Neighbours*.

Apesar da grande variedade de algoritmos existentes dentro desta categoria, aqueles que melhor se enquadram na temática deste projeto são as Árvores de Decisão, o *Naive Bayes* e o *K-Nearest Neighbours*, por isso mesmo, de seguida irão ser desenvolvidos cada um destes.

3.2.1 Árvores de Decisão

Uma árvore de decisão é definida como um procedimento que cria partições de um conjunto de dados recursivamente, criando divisões cada vez mais pequenas. A árvore é composta por uma raiz, vários nós e nós terminais, chamados folhas. Cada nó, numa árvore de decisão, tem apenas um nó parente e pode ter dois ou mais nós descendentes [17]. Numa árvore de decisão, é começado

o algoritmo a partir do topo da árvore e depois, é necessário seguir as ramificações corretas até obter a decisão a tomar, ou seja, até atingir o ultimo nó (folha).

Este método é muito útil devido à simplicidade de leitura, compreensibilidade, e facilidade no apoio à toma de decisões, por isso mesmo é um método bastante utilizado, sendo bastante útil nos sistemas de recomendação, como será explicado mais tarde.

3.2.2 *Naive Bayes*

O *Naive Bayes* é um método de *Machine Learning* que insere-se na família dos classificadores probabilísticos. Este algoritmo é altamente escalável e com uma razão entre complexidade e eficiência bastante positiva, o que o torna num método bastante popular.

Este método assume que cada valor de uma função particular é independente do valor de qualquer outra característica. Para além desta característica, este método tem como vantagem o facto de não necessitar de muita informação para tratar os dados para estimar os parâmetros necessários para a classificação.

Abstratamente, o *Naive Bayes* é um modelo que utiliza a probabilidade condicionada, ou seja, dado um problema para classificar, representado pelo vetor $x = (x_1, ..., x_n)$, ele atribui a sua probabilidade utilizando a equação $p(C_k \mid x_1, ..., x_n)$.

Dentro deste método existem outras vertentes, onde por exemplo se pode destacar o *Multinomial Naive Bayes*. Este modelo representa as frequências com que certos eventos são gerados, onde p_i é a probabilidade do evento i ocorrer. Sendo $x = (x_1, ..., x_n)$ um vetor da distribuição de resultados, e x_i a contagem do número de vezes que o evento i foi observado num determinado caso. De seguida está apresentado um exemplo [18] do algoritmo onde é possível observar a forma da sua implementação.

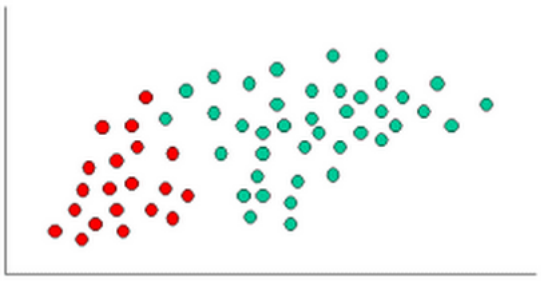

Figura 3.1: Exemplo *Naive Bayes* - Distribuição de Conjuntos

Como pode ser observado na figura 3.1, considerando os objetos como verdes ou vermelhos. O objetivo deste exemplo é prever qual será a classificação de um novo objeto no conjunto, utilizando apenas o conhecimento existente nos restantes objetos.

Tendo em conta que existem duas vezes mais objetos verdes que vermelhos, é sensato pensar que na existência de um novo objeto é duas vezes mais provável ser verde que vermelho. No *Naive*

Bayes esta crença é conhecida como *prior probability*. Estas são baseadas na experiência anterior, neste caso as percentagens dos objetos verdes e vermelhos, sendo muito usadas para prever eventos antes de estes acontecerem.

Nesta situação é possível concluir que:

$$\textit{Prior Probability para verde} = \frac{\textit{Número de objetos verdes}}{\textit{Número total de objetos}}$$

$$\textit{Prior Probability para vermelho} = \frac{\textit{Número de objetos vermelho}}{\textit{Número total de objetos}}$$

Assumindo que existem 60 objetos e que destes, 40 deles são verdes e 20 deles são vermelhos, então é possível calcular as *prior probabilities* dos conjuntos.

$$\textit{Prior Probability para verde} = \frac{40}{60}$$

$$\textit{Prior Probability para verde} = \frac{20}{60}$$

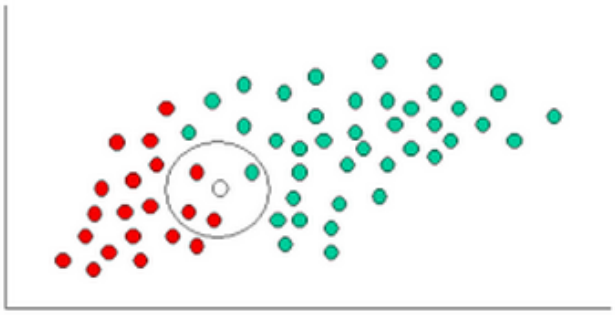

Figura 3.2: Exemplo *Naive Bayes* - Novo Objeto

Depois de ter sido calculada a *prior probability*, é possível classificar o nome objeto (circulo branco na figura 3.2). Assumindo que os objetos são incluídos corretamente num grupo, é possível assumir que a cor do novo objeto tem como maior probabilidade (*Likelihood*) a cor que for mais representada nesse grupo. Para medir essa probabilidade, é desenhado um círculo em torno de X que inclua um número de objetos definido à priori. Posteriormente é possível proceder ao cálculo do número de pontos dentro do círculo definido, utilizando as seguintes equações.

$$\textit{Likelihood para X verdes} = \frac{\textit{Número de objetos verdes na vizinhança de X}}{\textit{Número total de objetos verdes}} = \frac{1}{40}$$

$$\textit{Likelihood para X vermelhos} = \frac{\textit{Número de objetos vermelhos na vizinhança de X}}{\textit{Número total de objetos vermelhos}} = \frac{3}{20}$$

Apesar das *prior probabilities* indicarem que X é verde, a *likelihood* indica o contrário. Utilizando o método de *Naive Bayes*, a classificação desse novo objeto é feita tendo em conta ambas as características.

$$\textit{Probabilidade Posterior de X verde} = \textit{Prior probability verde} \times \textit{Likelihood de X verde}$$

$$\textit{Probabilidade Posterior de X verde} = \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

$$\textit{Probabilidade Posterior de X vermelho} = \textit{Prior probability vermelho} \times \textit{Likelihood de X vermelho}$$

$$\textit{Probabilidade Posterior de X vermelho} = \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Finalmente é possível concluir que o objeto X é vermelho pois possuí uma maior probabilidade posterior.

3.2.3 *K-Nearest Neighbours*

O *K-Nearest Neighbours* é um método de classificação que utiliza como entradas para o sistema os k exemplos mais próximos num grupo de características. Neste algoritmo, o *output* do sistema consiste no resultado obtido pela maioria dos resultados dos k vizinhos mais próximos, onde k é um inteiro positivo e que caso de $k = 1$, a saída do sistema é igual ao resultado do vizinho mais próximo.

Este encontra-se nos algoritmos de *Machine Learning* mais simples em termos de implementação, no entanto é bastante útil para atribuir pesos à contribuição de cada um dos vizinhos, de forma a que os vizinhos mais próximos possam ter um peso maior do que os restantes.

Para implementar o algoritmo, é necessário realizar o cálculo da semelhança entre utilizadores ou conjuntos, para isso podem ser utilizadas as fórmulas apresentadas em 2.3.1, de forma a obter a distância entre os conjuntos e assim obter os *k-nearest neighbours*.

3.3 Aprendizagem não Supervisionada

Tal como foi referido anteriormente, esta técnica tem como principal objetivo encontrar padrões existentes nas entradas do sistema e com isso, conseguir transformá-las nas saídas deste, sem o auxilio de um supervisor ou professor.

Considerando uma máquina que recebe uma sequência de entradas $x_1, x_2, x_3, ...$, onde x_t é a entrada no instante t. Estas entradas podem ser representações de uma imagem, som, ou por exemplo, uma lista de produtos num carrinho de supermercado.

Em sistemas não supervisionados, a informação recebida não contém nenhuma forma pré-estabelecida de como criar a saída. Por isso mesmo, o computador tem que desenvolver uma representação das entradas que possa ser utilizada nas predições, reconhecimentos ou até na tomada de decisões [19].

Os métodos de Aprendizagem não Supervisionada são muitas vezes confundidos e até associados com métodos de *Data Mining* que são utilizados para processar os dados. Existem alguns tipos de abordagens de aprendizagem não supervisionada, por exemplo:

- *Cluster Analysis* - consiste na organização de padrões em *clusters* baseados em proximidade [20].
- *Association Rule Learning* - é um método útil para descobrir relações entre variáveis em grandes bases de dados.

De seguida, irão ser apresentados os principais métodos de resolução de problemas dentro da aprendizagem não supervisionada, onde em cada uma das secções serão apresentados alguns dos algoritmos mais utilizados dentro de cada uma das categorias.

3.3.1 Cluster Analysis

Cluster Analysis, tal como Sarstedt [2] definiu, é um método para identificar grupos homogéneos de objetos, chamados *clusters*. Objetos dentro de um grupo partilham uma série de características, e são bastante diferentes de objetos de outros grupos [2].

Dado um problema de *cluster analysis*, as questões seguintes devem ser respondidas:

- Qual o objetivo de agrupar?
- Quais são as restrições a ser consideradas?
- Como deve ser feito o agrupamento?
- O agrupamento obtido é útil?

Para entender melhor como o método de *Cluster Analysis* funciona, é mais fácil seguir um exemplo, neste caso o que Sarstedt apresentou no seu livro "A Concise Guide to Market Research". Imaginando que o objetivo é analisar uma base de clientes para, por exemplo, uma questão de estratégia de preços. O primeiro passo é escolher as características que vão ser usadas para segmentar os utilizadores, ou seja, quais as variáveis que vão ser incluídas na análise. Neste caso, serão utilizadas como variáveis a lealdade à marca (*brand loyalty*) e a consciência de preço pelos consumidores (*Price Consciousness*). Estas variáveis são apresentadas de seguida, onde são avaliadas entre 1 e 7 e onde 7 corresponde a alta consciência de preço e alta fidelidade à marca.

De seguida será apresentada uma tabela e um gráfico que servirão de dados para exemplificar as abordagens das diferentes categorias de *Cluster Analysis*.

Tabela 3.1: Informação acerca dos clientes

Cliente	A	B	C	D	E	F	G
x	3	6	5	3	6	4	1
y	7	7	6	5	5	3	2

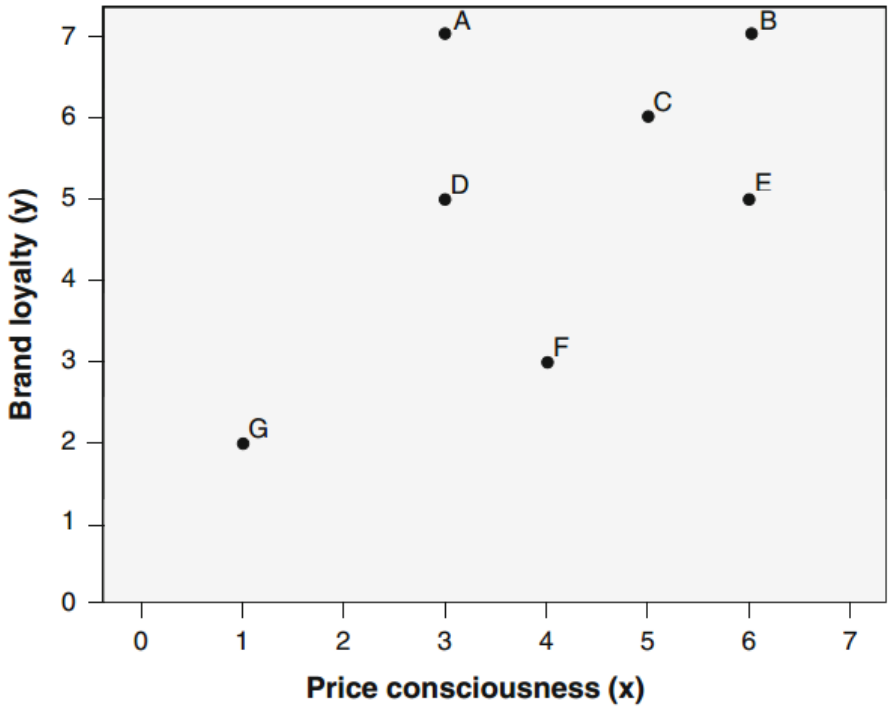

Figura 3.3: Distribuição das informações dos clientes [2]

Aqui é necessário decidir qual o procedimento a usar, porque existe mais do que uma abordagem possível para resolver este problema de agrupar tipos de produtos. As possibilidades que existem são as seguintes: Métodos Hierárquicos, Métodos de Divisão, e *Two-Step Clutering*.

Apesar de todos estes procedimentos possuírem o mesmo objetivo, criar *clusters* que partilhem informação, a proximidade entre os elementos de cada grupo tem de ser medida e por isso cada procedimento tem diferentes abordagens para a mesma questão, obtendo por isso mesmo resultados diferentes para o mesmo problema.

É então necessário decidir quantos grupos serão criados. Este número varia consoante o processo que é escolhido e, como é evidente, qual o problema que está a ser resolvido. Por exemplo, é diferente analisar um problema sobre a frequência de um cliente, ou por exemplo, que outro conjunto de produtos os clientes costumam comprar quando compram um determinado produto.

O número de *clusters* varia em cada problema e com cada abordagem. Nestas situações, com quantos mais grupos forem utilizados estes procedimentos, melhor são os resultados obtidos. No entanto, com o crescimento do número de grupo, a complexidade dos processos aumenta proporcionalmente.

Um passo muito importante a ser tomado é decidir quais serão as variáveis que irão ser a entrada do problema de *cluster analysis*. Aqui é necessário ter variáveis que sejam relacionáveis, no entanto se a proximidade entre elas for muito grande, isso vai provocar erros. Isto é, as variáveis têm que partilhar informação mas não devem ser iguais.

Depois de serem apresentadas as principais questões a serem resolvidas num problema de *Cluster Analysis*, vai ser apresentado de seguida, quais as principais abordagens em cada um dos métodos indicados anteriormente, métodos hierárquicos, particionais e *Two-Step Clustering*.

3.3.1.1 Métodos Hierárquicos

Depois de serem escolhidas as variáveis, é necessário escolher qual o procedimento de *clustering* a ser utilizado. Como foi explicado anteriormente, existem várias formas de resolver um problema de agrupamento, por isso vão ser apresentados inicialmente os métodos hierárquicos de *clustering*.

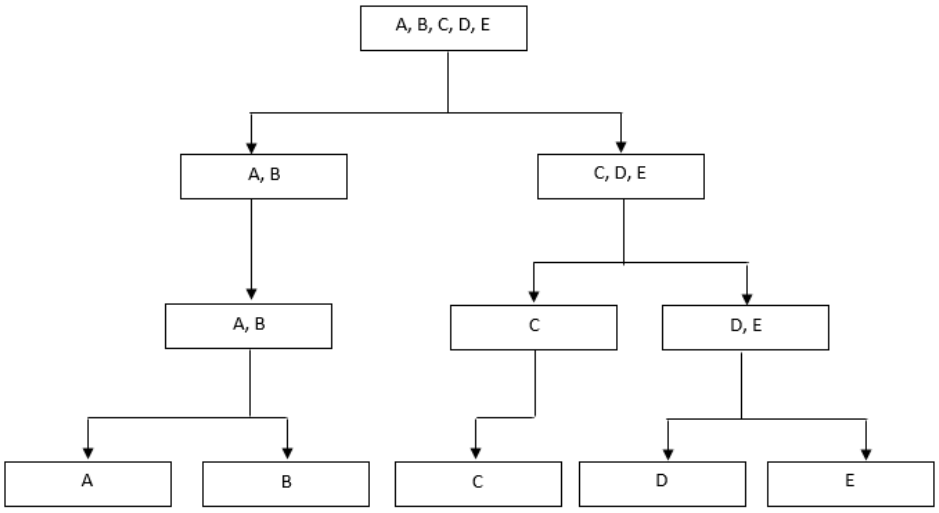

Figura 3.4: Exemplo utilizando o Método Hierárquico de *Clustering*

Os procedimentos hierárquicos são caracterizados por criar uma estrutura tipo árvore. Por vezes estes processos são confundidos com a categoria aglomerativa, onde cada grupo é formado a partir do objeto consecutivo. Nesta categoria, um par de *clusters* é fundido formando apenas um grupo em cada etapa, como pode ser visto na figura 3.4. No último passo, apenas resta um grupo, por isso não existe mais nenhuma ligação, então a aglomeração acaba e é formada uma árvore aglomerativa. Contudo esta hierarquia pode ser obtida por uma abordagem *top-down*, onde

do nível mais elevado descende pelo menos um filho até que o grupo já não possa ser subdividido mais nenhuma vez e a árvore fica completa.

O passo seguinte para resolver o problema é decidir qual o método a utilizar para calcular a semelhança ou diferença entre objetos. Existem vários métodos para utilizar mas uma das formas mais direta é medir a distância entre os pontos no gráfico 3.3. Neste caso, é apenas necessário medir a distância Euclidiana entre dois pontos. A equação seguinte apresenta o método de Euclides para medir a semelhança entre produtos.

$$d_{Euclidiana}(B,C) = \sqrt{(x_B - x_c)^2 + (y_B - y_C)^2} \tag{3.1}$$

Este método, calcula a distância que corresponde ao comprimento da linha que conecta dois pontos. A equação apresentada acima, funciona para sistemas de duas variáveis. No entanto, se o sistema utilizado tiver mais de duas variáveis, é necessário adicionar uma dimensão extra, ou seja, se no problema forem escolhidas três variáveis então a equação correta para calcular a semelhança é a seguinte:

$$d_{Euclidiana}(B,C) = \sqrt{(x_B - x_B)^2 + (y_B - y_C)^2 + (z_B - z_C)^2} \tag{3.2}$$

Com as equações apresentadas anteriormente, é possível obter a distância entre todos os pontos do gráfico ou objeto a partir dos dados. Por isso, é possível construir uma matriz com as distâncias, obtendo uma matriz com $(n+1) \times (n+1)$ células, onde n é o número de entradas, que neste caso em concreto corresponde a clientes.

Um dos problemas deste método é quando existem valores de variáveis que não têm os mesmos limites, ou seja, se a fidelidade à marca fosse medida de 1 a 7 e a consciência de preço de 1 a 15. Neste caso, seria necessário normalizar os valores para que assim, ambos utilizassem os mesmos limites e, posteriormente fosse possível utilizar a equação apresentada anteriormente.

Apesar de calcular distâncias entre pontos para variáveis medidas com valores ordinais, estes métodos podem ser problemáticos se os valores forem nominais. Neste caso, é necessário mudar de método, ou seja, mudar as variáveis para binárias. A tabela 3.2 é um exemplo desta mudança.

Tabela 3.2: Esquema para Correspondência de Coeficientes

		Objeto 2	
		Presença (1)	Ausência(0)
Objeto 1	Presença (1)	a	b
	Ausência (0)	c	d

Baseado nesta tabela podem ser computados diferentes *matching coefficients*, como por exemplo *Simple Matching Coefficient* (SM)

$$SM = \frac{a+d}{a+b+c+d} \tag{3.3}$$

Outros métodos são *Jaccard* (JC) e *Russel and Rao* (RR). A partir da tabela 3.2 podem ser obtidos da seguinte forma:

$$JC = \frac{a}{a+b+c} \tag{3.4}$$

$$RR = \frac{a}{a+b+c+d} \tag{3.5}$$

Apesar de existir mais do que uma técnica para combinar variáveis métricas, ordinais, ordinais e nominais, em todos os casos será perdida alguma informação que pode alterar o resultado da análise. Um dos principais conselhos é evitar juntar variáveis métricas com nominais no mesmo *cluster*.

Depois de escolher o método para calcular a distância ou semelhança, é necessário decidir qual algoritmo de *clustering* usar, existindo vários algoritmos e onde cada um deles possui as suas vantagens e desvantagens. Existem vários procedimentos aglomerativos e eles podem ser distinguidos pela forma como definem a distância entre grupos. Os procedimentos aglomerativos mais comuns são: *Single Linkage*, *Complete Linkage*, *Average Linkage* e por fim, *Centroid*.

No procedimento *Single Linkage*, a semelhança entre dois grupos é calculada pela distância entre os seus dois membros mais semelhantes [21]. Ou seja, a distância entre dois *clusters* é a distância mais curta entre dois membros em dois grupos. Este algoritmo tende a criar grupos maiores.

O algoritmo de *Complete Linkage* assume que a distância entre dois grupos é baseada na distância maior entre dois membros em dois grupos distintos. Este método, tem como grande problema valores discrepantes [22].

No método *Average Linkage*, como o próprio nome indica, a distância entre dois *clusters* é obtida pela média da distância entre todos os membros dos dois grupos.

Por último, no método *Centroid*, primeiro é calculado o meio do grupo e depois a distância entre grupos é medida através da distância do *centroid* do grupo.

Tanto o método *Average Linkage* como o método *Centroid* produzem grupos muito semelhantes e com poucas variações em termos de tamanho entre eles.

Em cada algoritmo apresentado, o procedimento é muito similar, diferindo apenas no ponto onde é realizada a medição. Contudo, depois de obtido esse ponto, os passos a seguir em todos os quatro métodos são semelhantes. Inicialmente é medida a distância entre todos os grupos e depois os *clusters* que têm distâncias iguais são aglomerados. O procedimento é repetido até obter apenas um grupo. A figura 3.5 é uma saída exemplo do sistema.

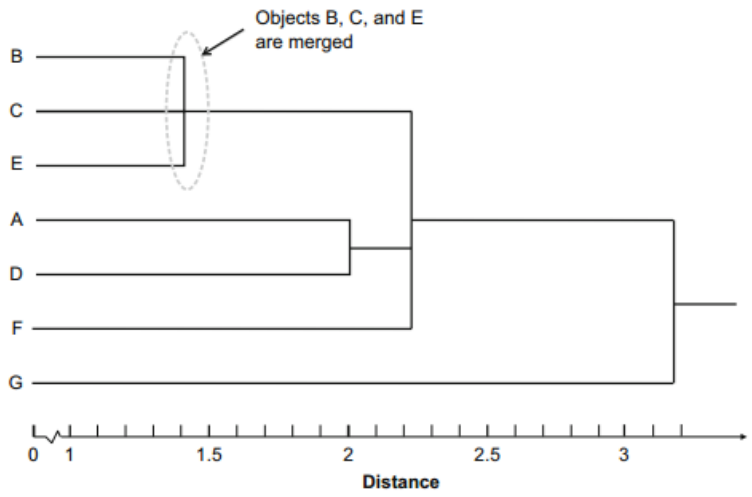

Figura 3.5: Resultado de um problema usando *Hierarchical Clustering*

3.3.1.2 Métodos Particionais

Outro grupo importante de procedimentos de *clustering* são os métodos particionais. Aqui, também existe uma vasta quantidade de técnicas diferentes. Contudo, o procedimento *k-means* é o mais utilizado e mais popular no mercado. Estes algoritmos minimizam um determinado critério de *clustering* realocando iterativamente pontos entre grupos até ser obtida a partição dos grupos ótima [23].

O algoritmo *k-means* é um método totalmente diferente dos métodos hierárquicos, e este segue um procedimento completamente distinto, onde a distância entre grupos não é medida, mas sim utiliza a variação dentro do *cluster* como como medida para formar grupos homogéneos.

Este tipo de algoritmo de *clustering* trabalha a partir de um número k de partições e o seu objetivo é descobrir qual as melhores k partições de n objetos [24].

Este algoritmo começa atribuindo aleatoriamente objetos aos k grupos especificados anteriormente. Estes objetos são constantemente reatribuídos a outros *clusters* de forma a conseguir minimizar a variação dentro dos grupos. Se a realocação de um objeto a outros *clusters* diminuir a variação dentro deste, estão o objeto é reatribuído a este *cluster*.

De seguida será continuado o exemplo referido anteriormente, agora para a apresentação do algoritmo *k-means*.

Passo 1: Usando o número k de *clusters*, é selecionado aleatoriamente o centro de cada *cluster*.

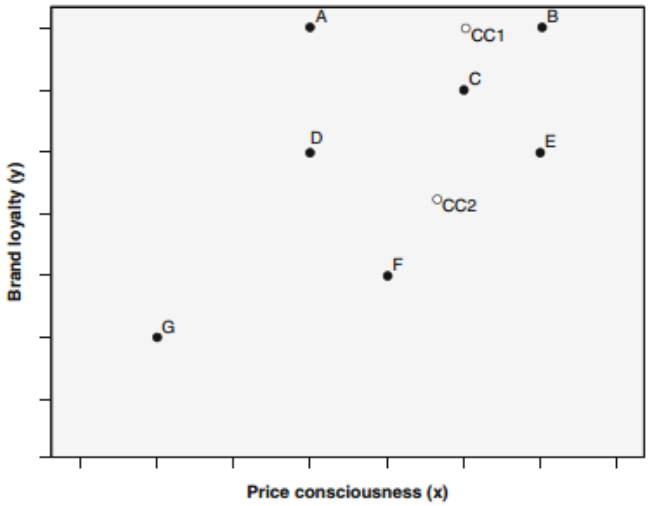

Figura 3.6: k-means passo 1

Passo 2: Aqui, são medidas as distâncias entre cada objeto até ao centro dos *clusters*. Depois de calculadas as distâncias Euclidianas, os objetos são atribuídos ao centro mais próximo, neste caso A, B e C são atribuídos ao primeiro grupo e D, E, F, G atribuídos ao segundo grupo.

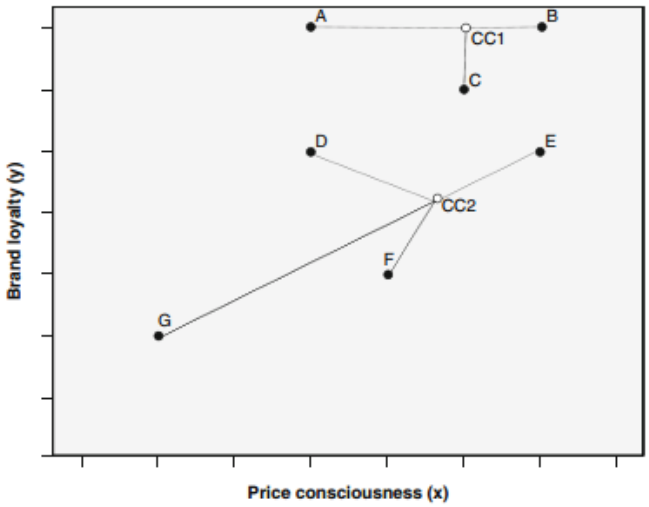

Figura 3.7: k-means passo 2

Passo 3: Baseada na partição anterior, o centro de cada grupo é medido, usando o valor médio dos valores dos seus objetos, relativamente às duas variáveis, neste caso consciência de preço e fidelidade à marca. Com isto, os centros dos grupos são movidos para as novas posições.

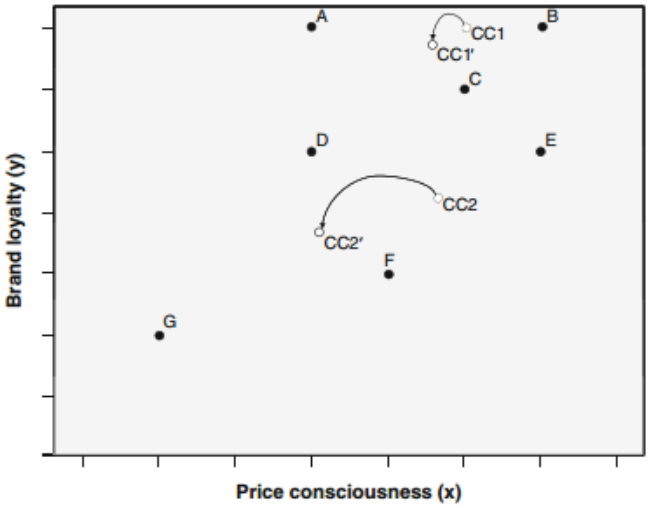

Figura 3.8: k-means passo 3

Passo 4: Com os novos centros dos *clusters*, alguns objetos podem ficar mais próximos de outros centros, por isso alguns objetos podem ser atribuídos a outros grupos.

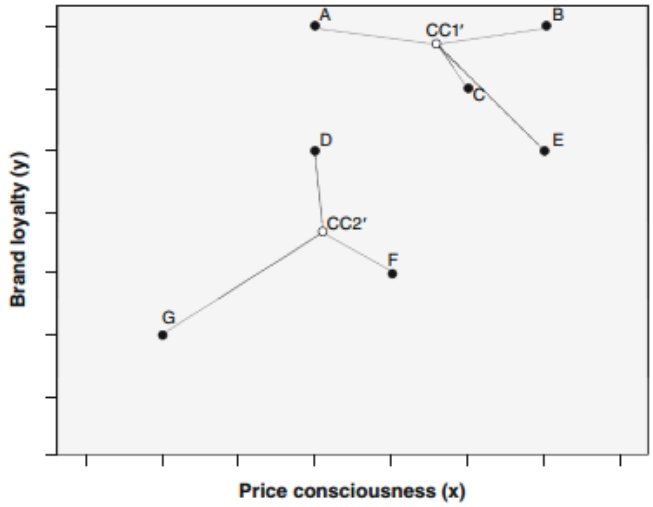

Figura 3.9: k-means passo 4

Posteriormente, o procedimento será repetido até um número predeterminado de vezes ser atingido ou ser obtida convergência nos resultados, ou seja, não existir mudanças de grupos por iteração.

3.3.1.3 Two-Step Clustering

Até agora, foram explicados dois dos três principais métodos de *clustering*. O método de *Two-Step Clustering* é o último deles e é um algoritmo desenhado para analisar grandes base de dados.

Este método, tal como o nome indica, é um algoritmo que se baseia em duas etapas. Primeiro, este algoritmo utiliza um procedimento muito semelhante ao, já explicado, algoritmo *k-means*. Depois desta primeira etapa, existe um passo com um *clustering* hierárquico aglomerativo modificado, método para formar *clusters* homogéneos.

Aqui, não é necessário predefinir os *k clusters* a formar, ou seja, este método determina automaticamente o número ótimo de grupos [25].

Neste método existe uma etapa de *pre-clustering*, este procedimento é implementado criando uma estrutura de dados chamada *Cluster Feature Tree* (CF), que contém os centros dos *clusters*. A árvore CF consiste em níveis de nós, onde cada nó tem um número de entradas. Para cada entrada, começando do nó de raiz, o nó filho mais próximo é descoberto recursivamente, descendo ao longo da árvore CF.

Depois desta etapa, é começado o passo de *clustering*, onde é desenvolvido o procedimento de *clustering* hierárquico aglomerativo, onde é descoberto o número ideal de *clusters*.

Este método pode trabalhar simultaneamente com variáveis categóricas e contínuas, e oferece ao utilizador a flexibilidade de especificar ou o número de *clusters*, ou o número máximo, ou até escolher automaticamente o número por avaliação estatística.

3.3.2 Association Rule Learning

Association Rule Learning tem como objetivo principal, descobrir todas as regras numa base de dados que satisfazem um mínimo de restrições de confiança [26]. Uma *association rule* é normalmente entendida como uma expressão na forma de

$$X \implies Y \tag{3.6}$$

Onde X e Y são conjuntos de variáveis em que $X \cap Y = \oslash$. Esta associação significa que uma determinada base de dados de transações D, onde em cada transação que contenha dados do conjunto X, provavelmente também contém dados do conjunto Y. A probabilidade ou *rule confidence* é definida pela percentagem de transações que contêm Y para além de X em relação ao total de transações contendo X.

Estes métodos foram estudados pela primeira vez no âmbito dos supermercados em 1993 por Agrawal [27]. Estes foram impulsionados pela introdução e evolução dos códigos de barras que proporcionaram o armazenamento de largas bases de dados acerca do histórico de compras dos utilizadores.

A ideia destes métodos, por exemplo em supermercados, é criar regras semelhantes à seguinte: "Um cliente que compra x_1 e x_2 também irá querer comprar o produto y com uma probabilidade de $c\%$". A sua utilidade numa empresa conjugada com a sua facilidade de entendimento faz com que a *Association Rule Learning* seja muito popular dentro dos métodos de *Machine Learning* e *Data Mining*.

Este método, normalmente enfrenta dois principais problemas: Primeiro, com o crescimento do número de produtos, a complexidade também aumenta, neste caso exponencialmente. Segundo,

facilmente podem ser geradas mais de 10 000 regras e, normalmente, as regras que realmente são úteis são uma pequena porção destas. Por isso mesmo, é muito difícil escolher e retirar regras interessantes [28].

Tal como referido anteriormente, numa implicação na forma $X \implies Y$, X é chamado de antecedente (*Ant*) e Y de consequente (*Suc*), onde a regra significa que X implica Y. Existem duas características básicas para *association rules*, que são a capacidade de suporte e o nível de confiança. A capacidade de suporte pode ser definida como:

$$sup = P(Ant \bigwedge Suc) = \frac{a}{a+b+c+d} \tag{3.7}$$

Supondo que o suporte de um produto é 0.1%, isto significa que apenas 0.1 por cento das transações contêm a compra deste produto [29].

O nível de confiança pode ser definido da forma seguinte

$$conf = P(Suc|Ant) = \frac{a}{a+b} \tag{3.8}$$

Supondo que a confiança na *association rule* $X \implies Y$ é de 80%, isto significa que 80% das transações que envolvem X também contêm Y [30].

Existem vários tipos de algoritmos de *Association Rule Learning*, no entanto aquele que se destaca e é mais popular é o algoritmo *Apriori*. Por isso mesmo, de seguida serão apresentadas as principais características de deste, vincando as suas maiores vantagens e desvantagens.

3.3.2.1 Algoritmo *Apriori*

Este algoritmo é desenhado para trabalhar com grandes base de dados que contêm transações, por exemplo as compras efetuadas pelos clientes ou até detalhes da utilização de um *website*. Cada transação é vista como um conjunto de itens. Dado um limite C, o algoritmo *Apriori* identifica os conjuntos de itens que são subconjuntos de pelo menos C operações na base de dados.

O algoritmo *Apriori* utiliza uma abordagem *bottom-up*, onde os sub-conjuntos são estendidos item a item (etapa conhecida como *candidate generation*) até não haver mais prolongamentos possíveis. Este algoritmo, procura conjuntos frequentes tendo em conta o suporte mínimo definido pelo utilizador. O pseudo-código para o algoritmo, apresentado por Agrawal e Srikant [31], está apresentado na figura 3.10. Este algoritmo é apresentado para uma base de dados de transações T e um limite de capacidade de suporte ε.

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)     C_k = apriori-gen(L_{k-1}); // New candidates
4)     forall transactions t ∈ D do begin
5)        C_t = subset(C_k, t); // Candidates contained in t
6)        forall candidates c ∈ C_t do
7)           c.count++;
8)     end
9)     L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ⋃_k L_k;
```

Figura 3.10: Pseudo-código do Algoritmo *Apriori*

Onde as variáveis seguintes correspondem a:

k - número de produtos num conjunto

L_k - conjunto de produtos com dois campos: produto e *support count*

C_k - conjunto de candidatos com dois campos: produto e *support count*

3.4 *Machine Learning* nas Apostas Desportivas

Depois de terem sido apresentadas as principais categorias de *Machine Learning*, os algoritmos mais representativos de cada uma delas e também os métodos mais indicados para o problema proposto, serão indicados os locais onde são utilizados os métodos de *Machine Learning* no mercado das apostas desportivas.

Desde o início dos desportos que as pessoas sempre tiveram o hábito de apostar sobre quais seriam os resultados dos mais variados eventos desportivos. No entanto, recentemente a indústria de apostas desportivas teve um crescimento enorme, tornando-se num mercado que movimenta milhões [32]. No entanto, a personalização no mercado das apostas desportivas não corresponde às necessidades atuais dos utilizadores. Para além disso, existem poucas publicações referentes à personalização dos mercados apresentados. Apesar disso, existem muitos projetos publicados no que toca à predição de probabilidades de vitória de cada equipa.

Existem muitos *papers* cujo tema é a predição de resultados, grande parte deles relacionados com futebol. São utilizados muitos procedimentos, desde *data mining* até métodos estatísticos. Uma abordagem muito comum neste tipo de questão é o uso das redes neuronais artificiais, onde com recurso aos dados estatísticos são construídas redes que ajudarão a calcular os resultados finais [33].

Para concluir, pode-se afirmar que apesar dos procedimentos de *Machine Learning* já existirem no mercado das apostas desportivas, estes, normalmente, são utilizados para calcular as probabilidades de vitória das equipas envolvidas e, nem tanto para adaptar o *website* aos hábitos de aposta dos utilizadores.

3.5 Resumo

Depois da pesquisa efetuada na área do *Machine Learning*, de conseguir analisar as principais metodologias e as categorias desta área, é possível tirar conclusões interessantes para o desenvolvimento do projeto. Por exemplo, quais os algoritmos com melhor ou pior razão entre a complexidade e eficiência e que será muito importante a utilização de dados de utilizadores semelhantes para conseguir solucionar o problema de utilizadores com pouco histórico, quer de apostas quer de visualização.

É também possível concluir que existem muitas abordagens possíveis para conseguir recomendar a cada utilizador jogos e/ou competições que possam estar interessados.

Verifica-se também que este é um projeto que vem de encontro às necessidades do mercado, que é uma área pouco explorada até então e que irá ser muito útil para uma melhor experiência de utilização do EMS.

Capítulo 4

Especificação e Implementação da Ferramenta

Neste capítulo será especificada qual a abordagem tomada para o desenvolvimento da solução encontrada e todos os detalhes da sua implementação. Primeiramente será apresentado o âmbito da ferramenta, onde será descrito o propósito da sua execução. Posteriormente, são apresentados os requisitos que foram tomados em consideração e que acabaram por ajudar a delinear a abordagem a tomar. De seguida será apresentada a arquitetura da ferramenta desenvolvida e onde é que esta se encontra na estrutura do EMS. Seguidamente, serão apresentadas as tecnologias utilizadas na implementação do projeto *Jarvis*. Depois, será apresentado todo o trabalho desenvolvido desde o estudo das possibilidades de implementação até à finalização do desenvolvimento da ferramenta. Finalmente, será feito um resumo de todos estes pontos onde serão apresentados os tópicos abordados e as suas conclusões.

4.1 Âmbito

Espera-se que a solução a desenvolver seja inovadora no ramo das apostas desportivas, oferecendo uma experiência focada no utilizador, onde serão destacados os eventos preferidos deste, recorrendo a sistemas de recomendação e a técnicas de *Machine Learning*. Esta ferramenta surge devido à necessidade crescente dos conteúdos na *Internet* serem adaptados às preferências dos utilizadores, onde o ramo das apostas desportivas não é exceção, apesar de até à data, não ser conhecido nenhum tipo de sistema semelhante na área.

A ideia geral do projeto *Jarvis* passa pela criação de um motor de recomendação que possibilite a adaptação do aspeto visual do EMS, que utilizando dados recolhidos anteriormente através da navegação e apostas dos utilizadores, forneça conteúdos personalizados, onde destaque a informação preferencial do utilizador, sem que este necessite de realizar qualquer tipo de configuração.

4.2 Requisitos

A ferramenta de recomendação a desenvolver necessita de cumprir alguns requisitos e funcionalidades para que esta se torne viável e útil. As principais funcionalidades desta são:

- Registar o comportamento e interação dos utilizadores: Para a utilização do motor de recomendação é necessário recorrer a dados acerca dos utilizadores, sejam estes visualizações ou apostas. É também necessário que este comportamento seja registado de forma automática, sem que o utilizador necessite de introduzir qualquer tipo de informação.

- Desenvolver o motor de recomendação: Para criar o sistema de recomendação é necessário criar um motor que recorrendo aos dados guardados, forneça os eventos preferidos do utilizador. É necessário implementar um algoritmo que calcule quais os eventos recomendados para o utilizador.

- Integração da ferramenta: A solução deve ser integrada dentro da estrutura da *Betfair* e do EMS.

- Adaptar o aspeto visual do EMS: O aspeto visual das páginas do EMS deverá ser adaptado utilizando os dados dos utilizadores e o motor de recomendação *Jarvis*.

- Eficiência: A ferramenta deverá ser eficiente e não sobrecarregar o EMS com os seus pedidos ou processamento pois será aplicada a um produto *mobile*.

- Solução flexível: A solução a implementar deverá ser flexível e escalável o suficiente de forma a que, sem alterações ao algoritmo, seja possível adicionar novas categorias de recomendação.

4.3 Arquitetura

No que toca à arquitetura, o EMS pode ser definido em três camadas, sendo estas o *Frontend* (FE), os Agregadores e os Serviços. Durante a utilização da aplicação, o FE comunica quer com os agregadores como com os serviços para obter os dados a utilizar por este, sendo que estes podem comunicar entre si para obter os dados necessários.

A ferramenta a desenvolver deverá ser implementada numa camada intermédia a estas, ou seja, entre o FE e os agregadores ou serviços, para poder comunicar com ambas as camadas e para além disso, integrar uma base de dados, estabelecendo ligações entre o FE e esta, bem como com o *Jarvis*.

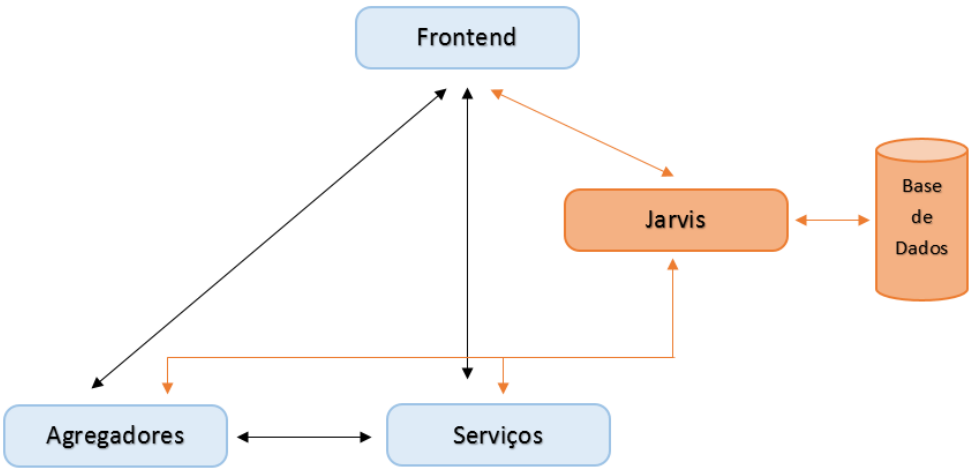

Figura 4.1: Arquitetura da Solução Proposta

Como é possível observar pela figura 4.1, que apresenta a arquitetura da solução, a aplicação *Jarvis* irá comunicar com a camada de *Frontend* da aplicação e os serviços ou agregadores de serviços. A aplicação irá fazer a ligação entre estes quando o *Jarvis* estiver ativo. Nesta situação, o FE fará pedidos ao *Jarvis*, consequentemente este fará pedidos à Base de Dados (DB), onde receberá informações sobre o histórico dos utilizadores, a aplicação criada comunicará então com os agregadores e os serviços para obter toda a informação para enviar para o *Frontend* e a personalização ser apresentada aos utilizadores.

4.4 Tecnologias

Para o desenvolvimento e implementação desta ferramenta, foram utilizadas várias tecnologias. Estas podem ser divididas nas seguintes áreas: *Frontend, Backend*, testes unitários e a base de dados.

No que toca ao *Frontend*, as tecnologias utilizadas foram *AngularJS* [34], *HTML* e *CSS*. O *Backend* da solução foi desenvolvido utilizando *Node.js* [35]. Os testes unitários utilizaram *Karma* [36] e *Jasmine* [37] para o *Frontend*, e *Chai* [38] e *Sinon.JS* [39] para o *Backend*. Devido ao facto da solução a desenvolver ter como objetivo ser integrada no EMS e ter sido desenvolvida desde o início com esse intuito, a escolha das tecnologias teve em conta essa situação e como tal, foi decidido escolher as tecnologias já utilizadas no EMS para facilitar a integração do *Jarvis*. Relativamente à base de dados, e visto que não existia nenhuma integrada no EMS, foi possível escolher a mais adequada. Depois de terem sido estudadas algumas base dados *noSQL* foi decidido utilizar *Redis* [40], devido à simplicidade de utilização, à variedade de comandos disponíveis na sua *API*, por ter suporte para *clusters* e sobretudo à sua velocidade de resposta, pois é desenvolvida para conjuntos do tipo *key-value* e possui uma memória *cache* com persistência.

Para além de todas as tecnologias utilizadas para a implementação da solução, todo o código desta seguiu certos padrões de qualidade, devido ao facto de ter sido integrada na aplicação do EMS, e assim sendo, foi necessário seguir as convenções deste com a utilização de *JSLint* [41] (ferramenta que é utilizada em desenvolvimento de *software* para validação de código *Javascript* seguindo um conjunto de regras de qualidade de implementação).

4.5 Trabalho Desenvolvido

Depois de serem definidos os objetivos da solução a implementar, foi necessário realizar um estudo acerca das ferramentas existentes e destas escolher aquelas que seriam uma boa base para o começo do desenvolvimento. Nesta fase, foram estabelecidos os requisitos que as ferramentas tinham de cumprir, para dessa forma poder ser selecionada qual a ferramenta a utilizar. Assim, foi definido que Escalabilidade, Eficiência, Ligação à Base de Dados, Facilidade de Manutenção do código, Facilidade de integração da solução e Adaptabilidade seriam os requisitos a cumprir pela ferramenta e assim seria escolhida a que cumprisse um maior número deles.

Inicialmente surgiram duas ferramentas que se destacavam das demais, o *PredictionIO* [42] e o *Raccoon* [43]. Posteriormente foi necessário proceder ao estudo de cada uma das ferramentas. Analisando-se ambas as soluções, concluiu-se que o *PredictionIO* tinha mais potencialidades, visto já ter incorporado ligação à base de dados *HBase* [44], um algoritmo de recomendação mais eficiente, um algoritmo adaptado à solução a implementar no EMS e um grande número de contribuições no projeto no *GitHub*, provando ser um projeto de interesse, com muitas pessoas envolvidas e em constante desenvolvimento.

Nesta fase, foi realizado um estudo mais aprofundado sobre o *PredictionIO* e posterior implementação da solução. Com o desenvolvimento desta, algumas desvantagens surgiram, nomeadamente em termos de integração no EMS, manutenção de código e sobretudo porque este não era muito eficiente em situações de muitos utilizadores e eventos, isto porque todos os produtos e utilizadores tinham de ser registados na base de dados e só depois poderiam ser criados os eventos entre os utilizadores e os produtos, o que produzia demasiadas entradas na base de dados sem nenhum efeito prático. Esta situação veio a agravar-se quando testada com grande número de utilizadores, eventos e produtos, por exemplo, quando testada com cerca de 5000 utilizadores, 10000 produtos e cerca de 20000 eventos.

Uma vez que as desvantagens do uso do *PredictionIO* pareciam cada vez maiores que as vantagens, foi altura de estudar o *Raccoon*, um motor de recomendação implementado em *Node.js*. Este utiliza como base de dados predefinida o *Redis* e é concebido para recomendar aos utilizadores filmes ou produtos, baseando-se no seu histórico de *likes* ou *dislikes*. Esta ferramenta apesar de ser bastante simples, possuir uma *API* bastante clara, ser escalável, ser de simples manutenção e integração no EMS, tinha como problema não se enquadrar no âmbito da ferramenta *Jarvis* a desenvolver, visto que este algoritmo está desenhado para classificar os produtos em gosto ou não gosto, no caso do EMS esses eventos não existem, apenas são registadas as visualizações e apostas, dessa forma, o algoritmo do *Raccoon* não se apresenta enquadrado com a solução a desenvolver.

Depois de terem sido estudadas estas ferramentas, foi colocada a hipótese de construir uma solução de raiz, desenvolvida em Javascript. Aqui, cedo foi possível concluir que esta poderia ser escalável, eficiente, adaptada à solução a desenvolver, com facilidade de integração e manutenção, no entanto, não possuía ligação já estabelecida à base de dados.

No final deste estudo foi possível chegar à tabela 4.1 que faz o resumo das soluções estudadas.

Tabela 4.1: Tabela de Comparação das Tecnologias

	PredictionIO	Raccoon	Solução em Javascript
Escalabilidade	✓	✓	✓
Eficiência	✗	✓	✓
Ligação à Base de Dados	✓	✓	✗
Facilidade de Manutenção do Código	✗	✓	✓
Facilidade de Integração da Solução	✗	✓	✓
Adaptabilidade à Solução a implementar	✓	✗	✓

Como é possível observar, através da comparação das tecnologias não foi possível concluir qual a decisão a tomar, pois tanto o *Raccoon* como a implementação em *Javascript* cumpriram com o mesmo número de requisitos. Para resolver esta situação foi decidido que a ligação à base de dados não tinha o mesmo grau de importância do que a adaptabilidade à solução a implementar no EMS, devido ao facto de que estabelecer a ligação não é tão complexo como adaptar a solução para o desejado. Assim, posteriormente a terem sido estudadas algumas ferramentas já existentes e de analisadas as suas virtudes e defeitos, foi possível definir que a abordagem a tomar seria um motor de recomendação desenhado de raiz em *Javascript*. Posteriormente foi definida qual seria a base de dados *noSQL* (não relacional) a utilizar, sendo que a decisão recaiu sobre o *Redis* tal como foi apresentado na secção 4.4.

4.5.1 Registo de Navegação e Apostas

Durante a utilização do EMS por parte de um utilizador, são despoletados eventos que registam a atividade deste, onde é arquivado o histórico da navegação, registando por exemplo, quais os desportos e os tipos de mercados visualizados e/ou apostados. Durante a utilização, os eventos despoletados incrementam o valor de cada ID na tabela da respetiva categoria referente ao utilizador em questão. Por exemplo, se o utilizador $u1$ carregasse no jogo *Newcastle vs West Ham* da lista da figura 4.2, seria incrementado o valor do desporto futebol na base de dados relativa ao utilizador $u1$, dentro da tabela de desportos.

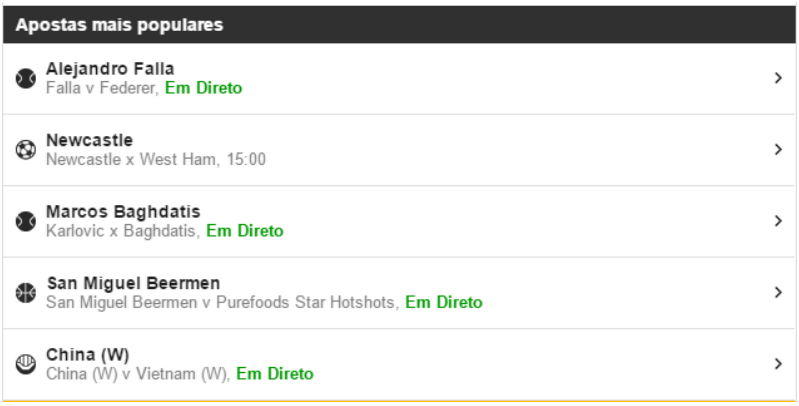

Figura 4.2: Exemplo de navegação no EMS

Para que os desportos e tipos de mercados fossem registados, foi necessário integrar na aplicação envios de informação para a base de dados em vários locais desta. Assim, durante a utilização do EMS são despoletados eventos que, através de um serviço desenvolvido para comunicar com o *Jarvis*, registam informações da utilização do apostador. Estes eventos foram distribuídos por muitos módulos, sendo eles *Most Popular Bets, Ribbon, Next Race* e *Side Menu* na página inicial (ver no anexo A), no módulo AZ foi adicionado no *Popular Sports* e no *All Sports* um evento que regista quais os desportos clicados, na página de um evento, no módulo *All Markets*, é registado o clique no mercado e desporto, e numa aposta é registado tanto o mercado como o desporto.

No caso do utilizador apostar nesse jogo o incremento é maior, ou seja, o peso da aposta é maior que o de uma simples visualização. Este peso é atribuído através de um ficheiro de configurações que atribui como aposta o peso de 5 visualizações, de forma a que uma aposta num evento tenha um maior impacto nas recomendações do que uma mera navegação. Ou seja, quando um utilizador navega para um evento, dá-se o incremento de uma unidade na contagem desse ID, no entanto, caso o utilizador aposte será somado à contagem deste 5 unidades.

4.5.2 Motor de Recomendação *Jarvis*

Quanto ao motor de recomendação, foi desenvolvido um sistema híbrido, que utiliza dois métodos de recomendação recorrendo à categoria *Weighted*, onde cada um dos métodos possui um peso na recomendação a realizar. A solução *Jarvis* foi implementada utilizando dois algoritmos, o *Naive Bayes* e o *K-Nearest Neighbours*. Aqui, o desenvolvimento pode ser divido em três fases, implementação do *Naive Bayes*, do *K-Nearest Neighbours* e do sistema híbrido com pesos.

4.5.2.1 *Naive Bayes*

Na solução desenvolvida, o algoritmo *Naive Bayes* descrito em 3.2.2 é utilizado para calcular as recomendações baseadas no histórico de cada utilizador, ou seja, para todos os utilizadores é calculada a recomendação através do seu histórico de navegação e de apostas.

Para o cálculo da recomendação é somado o número de contagens do utilizador dentro de uma categoria (por exemplo desporto), e depois, é calculada a probabilidade condicionada de cada um dos desportos existentes, calculando a razão entre a contagem do desporto e a contagem total.

4.5.2.2 *K-Nearest Neighbours*

O algoritmo *K-Nearest Neighbours* descrito em 3.2.3, é utilizado para calcular as recomendações para um utilizador tendo em conta as recomendações dos utilizadores mais próximos, onde *K* é definido inicialmente e a semelhança entre os utilizadores é obtida através da utilização do método *Jaccard Coefficient* descrito em 2.3.1.

Para calcular as recomendações começa-se por medir as semelhanças entre todos os utilizadores. Recorrendo ao método *Jaccard Coefficient* obtém-se um coeficiente entre 0 e 1 e que, quanto mais próximo de 1, maior é a semelhança entre ambos os utilizadores. Depois de calculados, é necessário reunir os *k* utilizadores mais semelhantes, obtendo as recomendações *Naive Bayes* de todos estes e atribuir ao utilizador estas recomendações.

4.5.2.3 *Weighted Hybrid Recommender System*

Depois de calculados os resultados de ambos os algoritmos, o motor de recomendação *Jarvis* atribui pesos a cada um dos algoritmos e assim constrói a recomendação final para cada utilizador.

Nesta fase, cada algoritmo tem a sua resposta e o *Jarvis* cria uma média ponderada entre ambos, onde nesta solução foi atribuída maior preponderância ao algoritmo *Naive Bayes*.

Os pesos atribuídos a estes procedimentos são definidos num ficheiro de configurações, que atribuem à método *Naive Bayes* um pesos de 70% e ao *K-Nearest Neighbours* 30%. A preponderância dos algoritmos foi assim definida para que as recomendações ao utilizador não fossem muito dispares daquilo que este mais aprecia apostar, no entanto foi definido que 30% das suas recomendações surgissem através do histórico dos seus "vizinhos", possibilitando desta forma, aumentar o leque de apostas e sugestões de mercados ou desporto.

4.5.3 Performance e Otimizações

Depois de ter sido desenvolvido o motor de recomendações, algumas alterações foram realizadas para otimizar os algoritmos desenvolvidos.

Inicialmente, o cálculo das recomendações foi alterado para utilizar uma técnica chamada *settle*, que consiste em separar todos os processos e realizar as conexões em paralelo, de forma a obter resultados mais rápidos. Esta técnica possibilita que caso um dos procedimentos falhe, todos os outros continuem o normal funcionamento e no fim, informe quais os procedimentos que correram corretamente ou não. Com isto, evita-se que na falha de algum processo, todos os outros deixem de correr.

Outra otimização realizada foi a alteração de todo o cálculo para ser processado em *chunks* de utilizadores, isto é, os cálculos eram realizados de 500 em 500 utilizadores, de forma a obter

maior segurança entre as ligações com a base de dados, maior velocidade nos cálculos e por fim, devido a limitações no número de conexões em paralelo à base de dados.

Para melhorar a performance do cálculo das semelhanças entre os utilizadores no procedimento do *K-Nearest Neighbours*, foi alterado modo de as calcular para, em vez de calcular a proximidade entre todos, passar a parar o calculo quando um destes já tiver nas suas semelhanças mais do que 10 apostadores com um coeficiente superior a 0.8, ou seja o procedimento para este utilizador é dado como finalizado. Assim, com esta alteração, a performance foi substancialmente melhorada, visto não necessitar de realizar comparações entre todos os utilizadores.

Por fim, depois de já terem sido realizados os testes do processo do *K-Nearest Neighbours*, foi percetível que a performance deste tinha de ser melhorada, como tal, foi decidido que o cálculo seria efetuado num sistema multi-processos, em que cada um deles simula correr em máquinas diferentes e desta forma melhorar substancialmente a duração deste. Para isso, o cálculo foi divido em vários processos, sendo que cada um deles calcula um *chunk* de cada vez, recorrendo a uma fila criada que guarda os grupos de utilizadores que ainda não foram processados e removendo desta o *chunk* que está a ser calculado, terminando o cálculo quando a fila estivesse vazia.

4.5.4 Processos Periódicos

Depois de concluído o motor de recomendação foi necessário definir como seriam calculadas as recomendações. Nesta altura algumas dúvidas surgiram quanto à abordagem a seguir, pois poderia ser calculada a recomendação em tempo real ou, por exemplo, calcular a recomendação uma vez por dia, registando os valores de recomendação numa tabela para cada utilizador, e fazer que esta não variasse durante o dia e apenas fosse atualizada durante a noite. Depois de ponderadas ambas as abordagens foi decidido criar um procedimento diário que pedisse ao motor de recomendação *Jarvis* para atualizar as recomendações de todos os utilizadores e depois, durante a utilização do EMS, apenas fossem realizados pedidos *GET* à base de dados, diminuindo o impacto do *Jarvis* no desempenho do *website*. Assim, o procedimento de cálculo das recomendações é efetuado uma vez por dia às 04:00h e começa removendo as entradas mais antigas da base de dados, depois calcula as recomendações *Naive Bayes*, as do algoritmo *K-Nearest Neighbours* e por fim, atualiza as recomendações pesadas para todos os utilizadores.

Nesta altura, para que apenas com uma visualização um desporto não passasse a ser recomendado, foi estabelecido um número mínimo de contagens (seja de visualizações ou apostas) para que a recomendação passasse a ser visível, ou seja, caso um desporto no algoritmo *Naive Bayes* tivesse uma contagem inferior a um número estabelecido como mínimo, por exemplo 20, a probabilidade desse desporto é 0, ou seja, não existiria recomendação nesse caso e os valores apresentados são os predefinidos, possibilitando ao *Jarvis* fornecer uma experiência mais coerente ao utilizador, não variando as recomendações com uma utilização muito reduzida.

Para além deste procedimento periódico estar definido, foi decidido que iria ser criado um processo de limpeza dos dados mais antigos, de forma a que a recomendação apenas se focasse nos últimos 180 dias de utilização (valor definido em ficheiro de configurações), de maneira a que fosse possível obter recomendações mais fiéis e também desta forma, reduzir o impacto de

eventos desportivos de grande dimensão, pois podem não representar a verdadeira preferência do apostador na utilização diária. Assim, todas as noites este procedimento é executado, verificando na base de dados todos os eventos mais antigos que o estipulado, removendo-os e decrementando o valor das contagens de cada utilizador na respetiva tabela.

4.5.5 Integração no EMS

Depois de desenvolvido o motor de recomendação *Jarvis*, foi necessário integrá-lo no EMS, onde foi criada uma configuração que ativava ou não o *Jarvis*, passando a partir daí a registar a navegação e a receber ou não recomendações durante a utilização. Para isso, bastava ir às configurações da aplicação e carregar no botão apresentado na figura 4.3, ativando ou não o *Jarvis*.

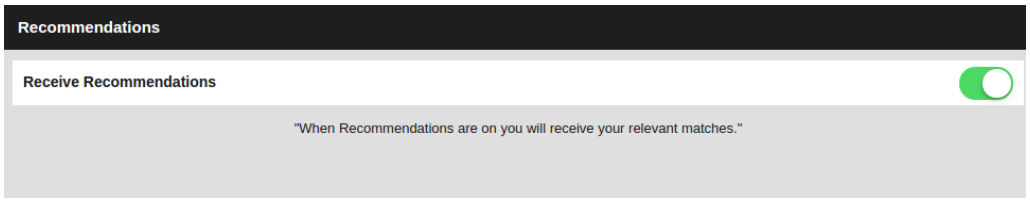

Figura 4.3: Botão de Ativação das Recomendações *Jarvis*

Aquando da integração do *Jarvis* no EMS, foram simulados históricos de utilizadores com tendências conhecidas, tal como será melhor explicado em 5.2, para dessa forma ir validando o correto funcionamento e integração das recomendações nos módulos.

Para que fossem integradas as recomendações, inicialmente foram alterados muitos dos pedidos efetuados pela aplicação a serviços externos, de forma a obter as recomendações do utilizador e posteriormente proceder às alterações desejadas nos módulos definidos. Assim, depois de terem sido efetuadas essas alterações na aplicação, foi possível proceder à alteração e criação dos módulos estipulados, *Popular Sports, My Popular Sports, Next Race* e *Serendipity*

4.5.5.1 *Popular Sports*

O primeiro a sofrer alterações foi o módulo *Popular Sports* no módulo *AZ*, onde os pedidos realizados foram alterados, caso tivesse a definição das recomendações ativa, de forma a obter os quatro desportos favoritos do utilizador, como se pode observar pelas figuras 4.4 e 4.5.

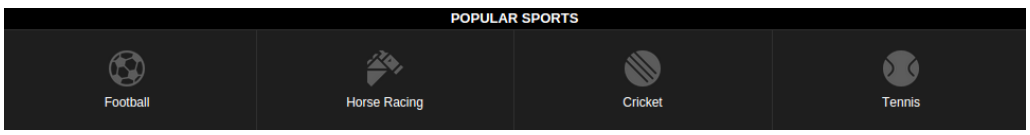

Figura 4.4: *Popular Sports* do Módulo *AZ* sem Recomendações *Jarvis*

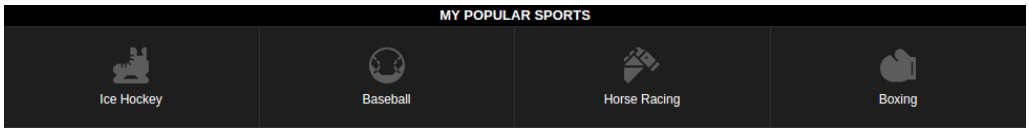

Figura 4.5: *Popular Sports* do Módulo *AZ* com Recomendações *Jarvis*

Como pode ser observado, na figura 4.4 são apresentados os desportos populares predefinidos e na figura 4.5 são apresentados os desportos favoritos do utilizador em questão. Caso o utilizador não possuísse quatro desportos na sua recomendação, os restantes eram preenchidos com os *popular sports* predefinidos. Neste módulo, quando as recomendações estão ativas, o nome é alterado, passando de *Popular Sports* para *My Popular Sports*.

4.5.5.2 *All Markets*

Posteriormente foi utilizada a recomendação do *Jarvis* no módulo *All Markets*, onde os mercados disponíveis num evento passaram a ser ordenados consoante a preferência do utilizador.

(a) *All Markets* sem Recomendações

(b) *All Markets* com Recomendações

Figura 4.6: Módulo *All Markets* sem e com recomendações do motor *Jarvis*

Como pode ser visto nas figuras apresentadas anteriormente, a figura 4.6a não possui nenhuma recomendação e como tal os mercados são apresentados consoante a ordem *default* e no

caso da figura 4.6b os mercados são apresentados de acordo com os gostos do utilizador. Apesar da personalização que é feita neste módulo, como é possível observar pelas figuras, o mercado *MATCH_ODDS* manteve-se na primeira posição. Devido a requisitos do produto foi definido que o primeiro mercado a apresentar seria sempre o predefinido, pois este é o mercado mais apostado e ajuda os utilizadores a definir as suas apostas nos restantes mercados.

4.5.5.3 *Most Popular Bets*

De seguida, foi adaptado o módulo *Most Popular Bets*. Este módulo para além de receber as apostas populares adaptadas ao gosto do utilizador mudou também de nome, passando a chamar-se *My Popular Bets* quando o motor de recomendação *Jarvis* está ativo.

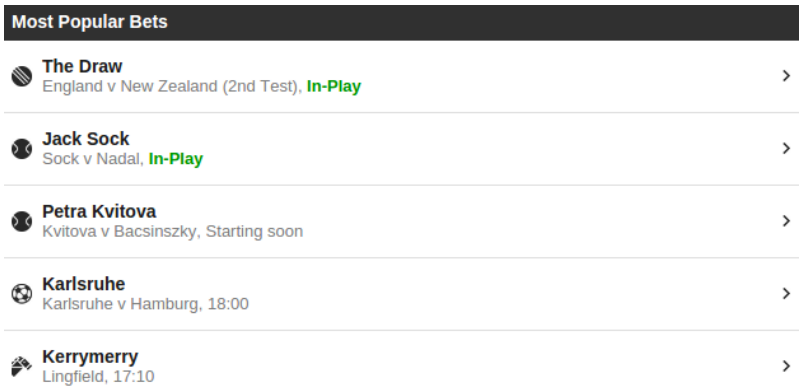

Figura 4.7: Módulo *Most Popular Bets* sem Recomendações *Jarvis*

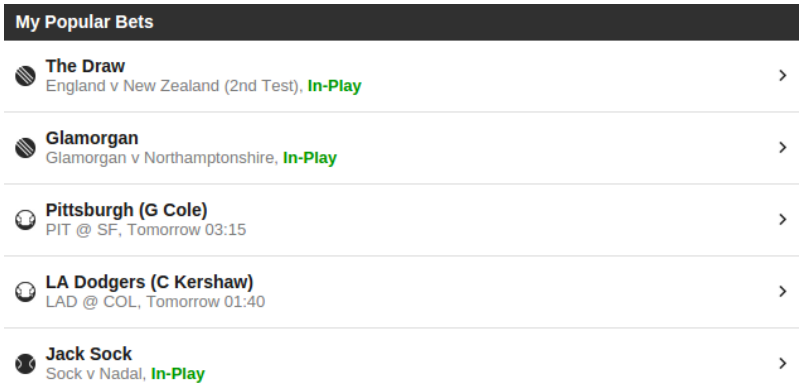

Figura 4.8: Módulo *Most Popular Bets* com Recomendações *Jarvis*

As figuras 4.7 e 4.8 são representações do módulo *Most Popular Bets* sem e com recomendações do motor *Jarvis*, respetivamente. Neste módulo, foi necessário proceder a algumas alterações no que toca aos pedidos realizados aos serviços ou agregadores destes. Aqui, inicialmente são pedidos ao *Jarvis* os cinco desportos mais recomendados ao utilizador, posteriormente, são pedidos

aos serviços, os eventos desses desportos que cumpram uma série de requisitos, nomeadamente liquidez, hora do evento, número de eventos por desporto, entre outros. Caso esses desportos não possuam cinco eventos que cumpram todos os requisitos, é feito um novo pedido para obter as apostas mais populares sem recomendação e estas são adicionadas às já obtidas pelo *Jarvis*.

4.5.5.4 *Next Race*

Seguidamente, apesar do módulo *Next Race* não ter sofrido nenhuma alteração a nível visual, este apenas está visível quando o utilizador tiver nas suas recomendações o desporto de corridas de cavalos, caso contrário, este módulo é escondido.

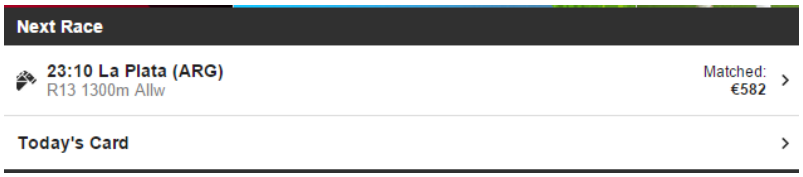

Figura 4.9: Módulo *Next Race*

Assim, o módulo apresentado na figura 4.9 tem de fazer um pedido ao serviço criado para comunicar com o motor de recomendação de forma a obter os desportos favoritos e depois, verificar se as corridas de cavalos estão presentes nas recomendações, para poder alterar a visibilidade do módulo.

4.5.5.5 *Serendipity*

Por fim, foi desenvolvido e integrado um módulo novo de seu nome *Serendipity* que tem como objetivo apresentar ao utilizador um evento com grande liquidez, de um desporto que não faz parte das recomendações calculadas para o utilizador, assim, é possível fazer com que o espectro de apostas do utilizador não diminua, mas pelo contrário, incentive o utilizador a apostar em novos desportos.

Figura 4.10: Módulo *Serendipity*

Para o módulo apresentado na figura 4.10, foi desenvolvido um algoritmo que obtém as recomendações do utilizador e posteriormente, obtém os eventos com maior liquidez dos desportos que não fazem parte dessa recomendação, de seguida é escolhido aleatoriamente um desses eventos, sendo que este é atualizado regularmente para que sejam sugeridos vários eventos.

4.5.6 EMS com *Jarvis*

Depois de terminada a integração do *Jarvis* no EMS, é possível verificar o impacto que este tem no EMS observando as figuras apresentadas no anexo A, onde é apresentado o aspeto visual do EMS com as recomendações desativas na secção A.1 e com as recomendações ativas na secção A.2.

4.6 Resumo

Neste capítulo, foram apresentados os requisitos e objetivos da solução a desenvolver, o motor de recomendação *Jarvis*, foi apresentada a arquitetura e as tecnologias e, por fim, o trabalho desenvolvido. Aqui, foi possível demonstrar quais as opções tomadas para o desenvolvimento do *Jarvis* e também para solucionar as dificuldades encontradas.

Na tabela 4.1 é possível observar o resumo das tecnologias estudadas e quais as vantagens e desvantagens de cada uma delas, bem como a razão para a escolha do *Javascript* como tecnologia para o desenvolvimento de uma solução implementada desde a raiz.

Seguidamente é possível observar quais as alterações efetuadas a nível de *Frontend* do EMS, sendo possível através das figuras apresentadas visualizar as alterações efetuadas tanto a nível de conteúdo como a nível visual.

Depois de todo este trabalho estar concluído, foi possível desenvolver um motor de recomendação e concluir que as escolhas das tecnologias foram acertadas e que estas se tornaram mais valias para o desenvolvimento e implementação do projeto.

Capítulo 5

Resultados Experimentais

No presente capítulo serão abordados os testes experimentais à ferramenta *Jarvis*. Estes foram efetuados em *Ubuntu 14.04* com um processador *Intel Core i5-2520M CPU 2.50GHz*.

5.1 Âmbito

Depois de terem sido apresentados todos os passos seguidos para a implementação e integração do *Jarvis* no EMS, foi necessário proceder aos testes funcionais e de desempenho para validar o trabalho desenvolvido e para concluir qual o real impacto da solução implementada na aplicação. Neste capítulo serão apresentados os testes especificados anteriormente e por fim, será apresentado um resumo do trabalho realizado bem como algumas das principais conclusões desta análise.

5.2 Testes de Funcionalidade

No que toca a testes de funcionalidade, para certificar que a funcionalidade da ferramenta era a esperada, foram desenvolvidos testes, recorrendo às tecnologias apresentadas em 4.4, e também testes exploratórios.

Os testes unitários são métodos de teste de software que testam pequenos blocos de código individualmente, fornecendo os dados de entrada e comparando os resultados da saída com os valores esperados [45].

Na ferramenta desenvolvida, todos os ficheiros foram submetidos a testes unitários de forma a certificar que cada trecho de código funcionava como esperado, comparando os resultados esperados de cada função com os resultados do *output* das mesmas.

Para além dos testes unitários, o *Jarvis* foi submetido a testes exploratórios. Para isso, foram criados vários utilizadores com tendência conhecida, geradas as recomendações e com a navegação na aplicação, foi verificado se os resultados das recomendações apresentados nos módulos correspondiam às tendências conhecidas dos utilizadores.

Para além destes testes, foram realizados testes para validação dos processos, de forma a verificar os valores obtidos por todos os processos que integram o motor de recomendação *Jarvis*.

Aqui, foram simuladas utilizações de vários apostadores em que cada um deles tinha um perfil conhecido, ou seja, era sabido à priori quais os desportos e mercados registados para cada um deles. Por exemplo, o histórico do utilizador $u1$ é apresentado na tabela 5.1.

Tabela 5.1: Histórico de Contagens do Utilizador u1

Ids de Desporto	Contagens
7524	100
7511	50
6	30
7	20

Ids de Mercado	Contagens
TO_SCORE	50
OVER_UNDER_25	30
OVER_UNDER_05	20
MATCH_ODDS	3

5.2.0.1 *Naive Bayes*

Inicialmente foi testado o primeiro dos processos periódicos, apresentado em 4.5.4, o cálculo das probabilidades *Naive Bayes*. Para verificar que este processo estava a ser realizado corretamente, as probabilidades de alguns utilizadores foram verificadas e assim, foi possível confirmar que os valores eram certos.

Tabela 5.2: Probabilidades do Utilizador u1

Ids de Desporto	Probabilidades
7524	0.50
7511	0.25
6	0.15
7	0.10

Ids de Mercado	Probabilidades
TO_SCORE	0.50
OVER_UNDER_25	0.30
OVER_UNDER_05	0.20

Como é possível concluir pela análise da tabela 5.2, o cálculo foi efetuado corretamente e como seria de esperar, contagens inferiores a 20 não interferem nas probabilidades, não sendo por isso recomendadas.

5.2.0.2 *K-Nearest Neighbours*

Seguidamente foi testado o processo de cálculo de semelhanças entre os utilizadores. Para isso, foram testados e avaliados os resultados da semelhança entre o utilizador $u1$ e $u10$, $u5$, $u20$ e $u15$. Nesta fase foi necessário recorrer às probabilidades e contagens dos utilizadores em questão, que são apresentadas nas tabelas B.1 para o $u10$, B.2 para o $u5$, B.3 para o $u20$ e B.4 para o $u15$, presentes no anexo B.

Com os valores apresentados anteriormente, foi possível calcular as semelhanças entre todos eles. Assim e como explicado em 2.3.1, é calculada a união e intersecção de entre estes.

Visto que os utilizadores $u1$ e $u10$ têm as mesmas probabilidades, tanto a intersecção como a união são iguais, ou seja, o coeficiente *jaccard* é 1.

Para os utilizadores $u1$ e $u5$, os valores da intersecção e união está presente nas tabelas 5.3 e 5.4 respetivamente.

Tabela 5.3: Intersecção entre os Utilizador u1 e u5

Ids de Desporto e Mercado	Probabilidades
7524	0.50
7511	0.25
6	0.10
1	0.10
TO_SCORE	0.50
OVER_UNDER_25	0.30
OVER_UNDER_05	0.20

Tabela 5.4: União entre os Utilizador u1 e u5

Ids de Desporto e Mercado	Probabilidades
7524	0.50
7511	0.25
6	0.15
1	0.15
TO_SCORE	0.50
OVER_UNDER_25	0.30
OVER_UNDER_05	0.20

Analisando os valores obtidos, e somando o valor das probabilidades destes, é possível calcular o *jaccard coefficient*.

$$jaccard_{u1-u5} = \frac{jaccard(desporto) + jaccard(mercado)}{2}$$

$$jaccard_{u1-u5} = \frac{\frac{0.5+0.25+0.10+0.10}{0.5+0.25+0.15+0.15} + \frac{0.5+0.3+0.2}{0.5+0.3+0.2}}{2} = \frac{0.9+1}{2} = 0.95$$

De seguida, seguiu-se o mesmo processo para os utilizadores $u1$ e $u20$, sendo que os valores da intersecção e união destes são apresentados nas tabelas 5.5 e 5.6 respetivamente.

Tabela 5.5: Intersecção entre os Utilizador u1 e u20

Ids de Desporto e Mercado	Probabilidades
7524	0.50
7511	0.25
7	0.10
TO_SCORE	0.50
OVER_UNDER_25	0.30
OVER_UNDER_05	0.20

Tabela 5.6: União entre os Utilizador u1 e u20

Ids de Desporto e Mercado	Probabilidades
7524	0.50
7511	0.25
6	0.15
7	0.15
7522	0.10
TO_SCORE	0.50
OVER_UNDER_25	0.30
OVER_UNDER_05	0.20

Com os valores apresentados nas tabelas e somando o valor das probabilidades destes, é possível calcular a semelhança destes utilizadores.

$$jaccard_{u1-u20} = \frac{jaccard(desporto) + jaccard(mercado)}{2}$$

$$jaccard_{u1-u20} = \frac{\frac{0.5+0.25+0.10}{0.5+0.25+0.15+0.15+0.10} + \frac{0.5+0.3+0.2}{0.5+0.3+0.2}}{2} = \frac{0.739+1}{2} = 0.869$$

Por fim, foi calculado o coeficiente de semelhança entre os utilizadores $u1$ e $u15$, sendo que os valores da intersecção estão apresentados na tabela 5.7 e da união na tabela 5.8.

Tabela 5.7: Intersecção entre os Utilizador u1 e u15

Ids de Desporto e Mercado	Probabilidades
7524	0.50
7511	0.25
6	0.15
TO_SCORE	0.50
OVER_UNDER_25	0.30

Tabela 5.8: União entre os Utilizador u1 e u15

Ids de Desporto e Mercado	Probabilidades
7524	0.50
7511	0.25
6	0.15
1	0.10
7	0.10
TO_SCORE	0.50
OVER_UNDER_25	0.30
OVER_UNDER_05	0.20
MATCH_OODS	0.20

Recorrendo às probabilidades da intersecção e união é possível calcular a semelhança entre os utilizadores.

$$jaccard_{u1-u15} = \frac{jaccard(desporto) + jaccard(mercado)}{2}$$

$$jaccard_{u1-u15} = \frac{\frac{0.5+0.25+0.15}{0.5+0.25+0.15+0.10+0.10} + \frac{0.5+0.3}{0.5+0.3+0.2+0.2}}{2} = \frac{0.818+0.667}{2} = 0.742$$

Depois de obtidos os coeficientes *jaccard* calculados anteriormente, foi necessário verificar nas tabelas de semelhança do utilizador $u1$ (tabela na qual é registada a semelhança entre o $u1$ e os restantes utilizadores) se os valores guardados correspondiam aos agora calculados para confirmar que os cálculos estavam a ser correctamente realizados.

Tabela 5.9: Coeficientes de *Jaccard* para o Utilizador u1

Utilizadores	Coeficientes
10	1.000
5	0.950
20	0.869
15	0.742

Como é possível concluir pela análise da tabela 5.9, os valores dos coeficientes de semelhança obtidos correspondem aos valores calculados anteriormente.

5.2.0.3 *Weighted Hybrid Recommender System*

Por fim, foi necessário verificar se o cálculo das recomendações estava ou não a ser bem calculado. Para isso, foram utilizadas as simulações anteriores, recorrendo ao mesmo número de utilizadores, histórico de contagens e semelhanças.

Para testar este procedimento, inicialmente foi verificado quais dos utilizadores calculados têm uma semelhança maior que 0.8 (valor estabelecido como semelhança mínima para ser dado como *neighbour*), no qual resultaram os utilizadores $u10$, $u5$ e $u20$.

Seguidamente, começaram a ser calculados os valores das recomendações recorrendo ao método apresentado em 4.5.2.3, denominado de *Weighted Hybrid Recommender System*. Nesta fase, recorreu-se às tabelas de probabilidade de cada utilizador (disponível no anexo B) para calcular as probabilidades da recomendação ao $u1$.

$$Probabilidade(7524) = 0.8 * 0.5 + 0.2 * \frac{0.5}{3} * 3 = 0.50$$

$$Probabilidade(7511) = 0.8 * 0.25 + 0.2 * \frac{0.25}{3} * 3 = 0.25$$

$$Probabilidade(6) = 0.8 * 0.15 + 0.2 * (\frac{0.15}{3} + \frac{0.10}{3}) = 0.1367$$

$$Probabilidade(7) = 0.8 * 0.10 + 0.2 * (\frac{0.10}{3} + \frac{0.15}{3} + \frac{0.15}{3}) = 0.1067$$

$$Probabilidade(7522) = 0.8 * 0 + 0.2 * \frac{0.10}{3} = 0.0067$$

Como as probabilidades dos IDs de mercado são iguais em todos os utilizadores, o valor da recomendação não sofre nenhuma alteração.

Tabela 5.10: Recomendações para o Utilizador u1

Ids de Desporto e Mercado	Probabilidades
7524	0.50
7511	0.25
6	0.1367
7	0.1067
7522	0.0067
TO_SCORE	0.50
OVER_UNDER_25	0.30
OVER_UNDER_05	0.20

Depois de efetuados todos os cálculos, foi possível verificar pela a análise da tabela 5.10, relativa às recomendações para o utilizador $u1$, que os valores calculados correspondem aos obtidos pelo motor de recomendação *Jarvis*.

5.3 Testes de Desempenho

Para testar a ferramenta desenvolvida em termos de performance, foram desenvolvidos vários testes para poder analisar o verdadeiro impacto que o *Jarvis* tem na atual aplicação do EMS. Para isso, foram desenvolvidos testes ao cálculo das recomendações, à limpeza de dados antigos e por fim, aos pedidos efetuados aos serviços para obtenção dos dados, apresentando os valores com e sem o *Jarvis* ativo, para assim analisar o real impacto deste.

5.3.1 Motor de Recomendação

Os primeiros testes a serem desenvolvidos foram ao Motor de Recomendação *Jarvis*. Aqui, foram testados os três procedimentos que fazem parte deste (apresentado na secção 4.5.2), ou seja, o cálculo das probabilidades do algoritmo *Naive Bayes*, o cálculo das semelhanças entre os utilizadores do algoritmo *K-Nearest Neighbours* e por fim, o cálculo das recomendações, recorrendo ao método *Weighted*.

Para poder analisar a performance destes procedimentos, foram simulados cálculos para 500, 1000, 5000, 10000 e 20000 utilizadores, onde cada um deles tinha no seu histórico 20 tipos de mercados e 20 desportos. Para gerar estes dados, foi desenvolvido um *script* que introduz visualizações de forma a que os utilizadores não possuam históricos semelhantes e assim consiga calcular as semelhanças entre eles de forma mais real.

5.3.1.1 *Naive Bayes*

Para analisar este procedimento, foram realizados testes nas condições apresentadas anteriormente e foram obtidos os resultados apresentados na tabela 5.11.

Tabela 5.11: Resultados do Cálculo do *Naive Bayes*

	Número de Utilizadores				
Tempos (T)	500	1000	5000	10000	20000
T1 (s)	0.437	0.845	4.459	8.231	16.324
T2 (s)	0.412	1.059	5.010	7.530	15.323
T3 (s)	0.445	1.015	3.596	8.145	19.624
T4 (s)	0.486	0.994	4.347	7.849	15.538
T5 (s)	0.447	1.013	4.025	8.366	14.337
Média (s)	0.445	0.985	4.287	8.024	16.229

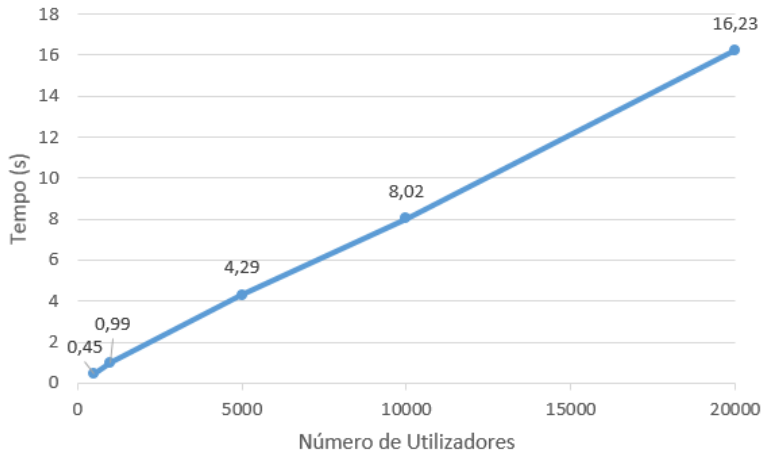

Figura 5.1: Valores obtidos pela média dos testes de cálculo do algoritmo *Naive Bayes*

Como é possível observar pela tabela 5.11, o cálculo do algoritmo *Naive Bayes* é bastante rápido, conseguindo obter resultados bastante positivos para 20000 utilizadores, 20 mercados e 20 desportos. Para além disso, é possível concluir pela análise da figura 5.1 que o tempo despendido neste procedimento cresce linearmente com o número de utilizadores.

5.3.1.2 *K-Nearest Neighbours*

De seguida, foi testado o algoritmo *K-Nearest Neighbours*, responsável pelo cálculo da semelhança entre todos os utilizadores. De seguida são apresentados os resultados obtidos relativamente ao tempo necessário para a realização desta tarefa.

Tabela 5.12: Resultados do Cálculo do *K-Nearest Neighbours*

	Número de Utilizadores				
Tempos (T)	500	1000	5000	10000	20000
T1 (min)	1.710	8.141	11.103	47.748	307.93
T2 (min)	1.655	7.554	11.454	47.039	302.466
T3 (min)	1.921	7.655	10.535	56.811	286.090
T4 (min)	1.7812	7.899	11.403	48.676	296.117
T5 (min)	1.694	7.772	10.033	48.109	279.129
Média (min)	1.759	7.805	10.846	49.675	294.347

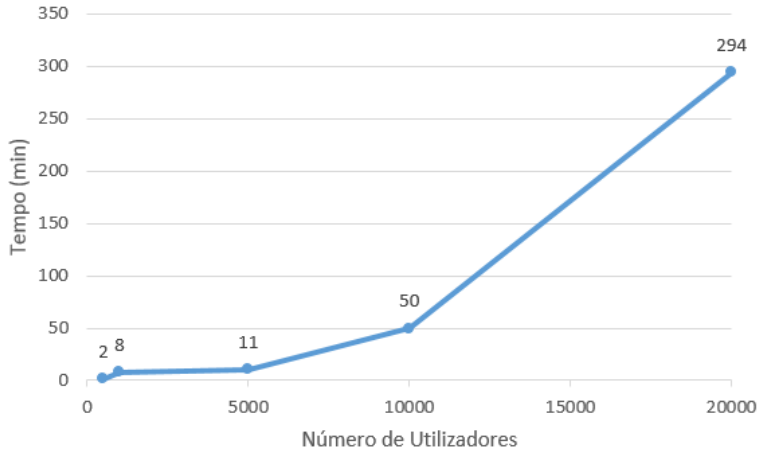

Figura 5.2: Valores obtidos pela média dos testes de cálculo do algoritmo *K-Nearest Neighbours*

Depois de analisar os resultados apresentados pela tabela 5.12, é possível concluir que este procedimento é muito mais moroso que o apresentado em 5.3.1.1, porque apesar deste também ser dependente do número de utilizadores presentes na base de dados, a duração deste cálculo cresce quase quadraticamente com o número de utilizadores, como é possível observar na figura 5.2. Apesar desta relação não ser a mais indicada, pois é desejada uma solução escalável com o número de utilizadores, é possível observar pelo gráfico que a tendência não é completamente exponencial, visto que o algoritmo foi um pouco alterado para a melhoria de performance, pois em vez de cada utilizador ser comparado com todos os existentes na base de dados, este apenas faz a comparação até encontrar o número de utilizadores semelhantes estipulado inicialmente.

Como foi apresentado na secção 4.5.3, relativa à performance e otimizações do algoritmo, algumas alterações foram realizadas para diminuir o tempo de duração deste procedimento, visto que este é por larga margem o mais moroso, como tal, foi desenvolvido um processo que paralelize o cálculo das semelhanças. Depois de concluída esta alteração, foram realizadas medições nas mesmas condições das anteriores, no entanto com o procedimento a correr em 6 processos separados, simulando um procedimento multi-máquina.

Tabela 5.13: Resultados do Cálculo do *K-Nearest Neighbours* com processo Paralelizado

	Número de Utilizadores				
Tempos (T)	500	1000	5000	10000	20000
T1 (min)	1.45	3.26	3.37	13.23	39.23
T2 (min)	1.10	3.77	3.93	4.36	42.38
T3 (min)	1.06	3.58	3.74	9.42	42.55
T4 (min)	1.01	3.65	3.73	7.05	44.65
T5 (min)	1.06	3.60	4.08	6.95	42.43
Média (min)	1.1	3.6	3.8	8.2	39.2

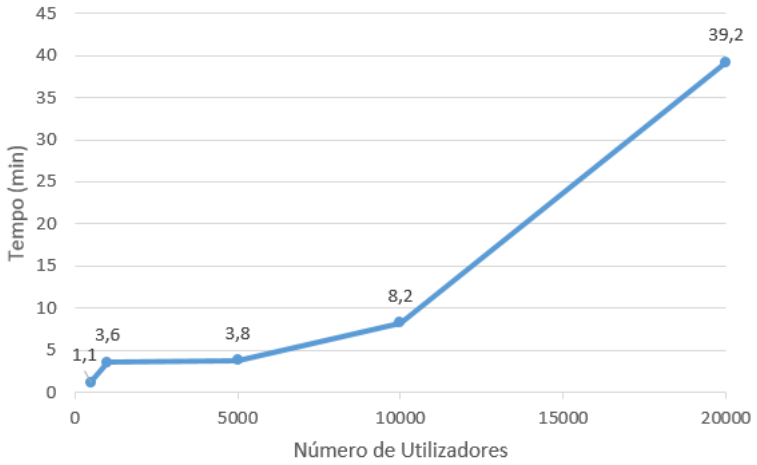

Figura 5.3: Valores obtidos pela média dos testes de cálculo do algoritmo *K-Nearest Neighbours* com processo Paralelizado

Depois de efetuados os testes ao novo método de cálculo do *K-Nearest Neighbours*, verifica-se que a performance melhorou significativamente, cerca de 7 vezes mais rápida. A razão para esta melhoria está assente no facto deste cálculo ter sido divido em 6 máquinas, em que cada uma delas processa um *chunk* de cada vez e desta forma, é possível obter um resultado muito mais rápido e que demonstra que a solução desenvolvida é escalável e preparada para ser utilizada no contexto do *Exchange Mobile Site* da *Betfair*.

5.3.1.3 *Weighted Hybrid Recommender System*

O último método a calcular dentro do processo de geração das recomendações *Jarvis* é o responsável pela média pesada dos dois procedimentos calculados anteriormente. Aqui, tal como nos restantes processos, foram realizados testes nas mesmas circunstâncias, de forma a concluir qual o tempo necessário para realizar o cálculo das *Weighted Recommendations*.

Tabela 5.14: Resultados do Cálculo das *Weighted Recommendations*

	Número de Utilizadores				
Tempos (T)	500	1000	5000	10000	20000
T1 (s)	0.817	2.344	7.617	13.629	27.517
T2 (s)	0.986	2.582	6.035	14.296	28.533
T3 (s)	0.959	4.589	6.953	13.421	27.470
T4 (s)	0.856	2.649	5.674	13.969	28.194
T5 (s)	1.018	2.499	5.655	13.719	26.394
Média (s)	0.927	2.932	6.386	13.806	27.621

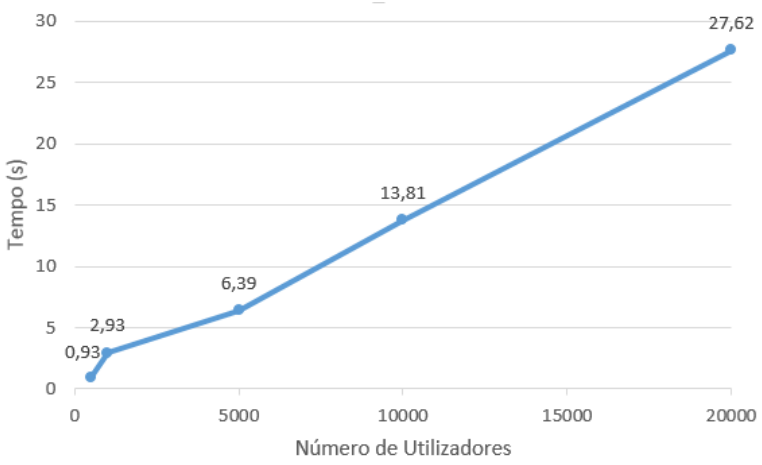

Figura 5.4: Valores obtidos pela média dos testes de cálculo das *Weighted Recommendations*

Observando a tabela 5.14 o tempo de cálculo deste procedimento é bastante baixo e tal como o método de cálculo das probabilidades é linearmente proporcional ao número dos utilizadores, como é possível concluir pela análise da figura 5.4.

5.3.1.4 Resultados Totais

Para concluir os testes realizados ao motor de recomendações, foram realizados testes ao cálculo completo destas, ou seja, desde que o era despoletado o cálculo das recomendações para os utilizadores, até que estas estavam concluídas. De seguida estão apresentados os valores obtidos pela média dos tempos de duração do cálculo das recomendações.

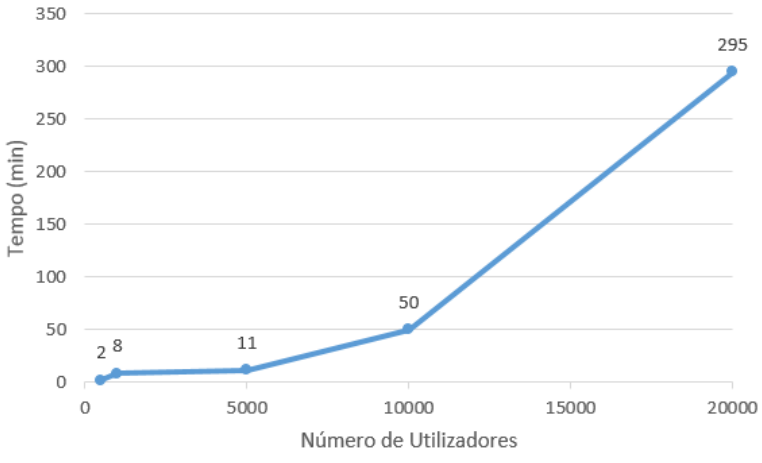

Figura 5.5: Valores obtidos pela média dos testes para a geração das recomendações

Depois de analisar a figura 5.5, é possível concluir que o tempo de cálculo das recomendações tem um crescimento quase quadrático com o número de utilizadores e o procedimento que tem

maior impacto no cálculo total das recomendações é o processo do cálculo dos *K-Nearest Neighbours*. Apesar disso mesmo, é possível concluir que o tempo de cálculo de todo este processo cumpre os objetivos traçados, pois consegue realizar todo o procedimento em cerca de 5 horas para 20000 utilizadores, sobretudo porque este é apenas feito uma vez por dia.

5.3.2 Limpeza de Dados Antigos

Outro dos processos cronológicos efetuados é o responsável pela limpeza dos dados antigos. Aqui, o tipo de testes foi significativamente diferente, ou seja, foram realizados testes com 40000, 60000, 80000, 100000, 120000 e 150000 em cada um dos dias a efetuar limpeza.

Tabela 5.15: Resultados das medições da Limpeza de Dados Antigos

	Número de Mercados					
Tempos (T)	40000	60000	80000	100000	120000	150000
T1 (s)	35.0	65.3	69.2	83.5	125.9	175.6
T2 (s)	30.7	62.4	63.3	81.2	131.4	184.2
T3 (s)	43.7	64.5	63.6	79.3	137.8	139.2
T4 (s)	44.3	52.2	65.9	80.3	129.8	141.1
T5 (s)	48.2	60.1	65.3	80.0	127.7	128.1
Média (s)	40.4	60.9	65.5	80.9	125.9	175.6

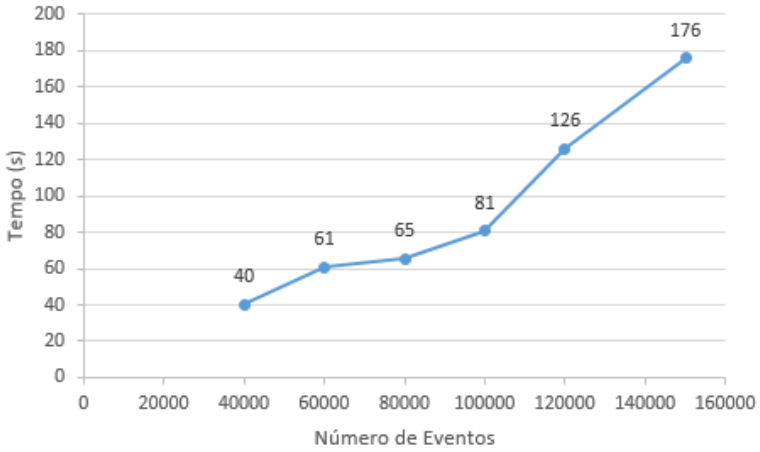

Figura 5.6: Valores obtidos pela média dos testes realizados na Limpeza de Dados Antigos

Depois de realizados os testes e analisados os resultados obtidos na tabela 5.15 é possível concluir que este é um procedimento rápido e que não tem impacto na performance do *Jarvis*. A duração deste procedimento cresce com o número de eventos registados na base de dados no dia especificado, ou seja, com o crescimento do número de eventos, o tempo da limpeza dos dados aumenta quase linearmente, sendo estes eventos dependentes do número de utilizadores, mercados ou desportos, como é possível verificar no gráfico apresentado na figura 5.6.

5.3.3 Módulos ou Serviços

Depois de completados os testes à ferramenta desenvolvida, foi necessário desenvolver os testes aos módulos e serviços para perceber qual o real impacto das recomendações *Jarvis* no EMS, ou seja, qual o tempo de resposta da aplicação com ou sem as recomendações ativas. Para isso, foram realizados pedidos ao EMS e foi avaliado qual o tempo de resposta de cada um deles, para poder analisar o verdadeiro impacto deste.

Para avaliar e retirar verdadeiras ilações da sua performance, foi avaliada a performance dos módulos adicionados e serviços modificados, nomeadamente nos módulos *Popular Bets, All Markets, AZ, Serendipity* e no serviço criado para comunicar com a base de dados do *Jarvis*, o *Top Recommendations*.

5.3.3.1 Popular Bets

O primeiro módulo a ser avaliado foi o *Popular Bets*, para isso foram realizados 20 testes (no anexo C.1) com as recomendações ativas ou não. Este é um dos módulos com maior lógica associada e o tempo de resposta depende de alguns fatores, nomeadamente, possuir um número suficiente de desportos nas suas recomendações, eventos com liquidez suficiente, dentro das categorias favoritas, entre outros. Os testes seguintes foram desenvolvidos para o pior dos casos, ou seja, os eventos recomendados não serem suficientes e por isso, ter que fazer novos pedidos para preencher o módulo com os eventos *default*.

Tabela 5.16: Resultados do Impacto do *Jarvis* no módulo *Popular Bets*

	Popular Bets	
	sem *Jarvis*	com *Jarvis*
T1 (ms)	203	507
T2 (ms)	208	392
T3 (ms)	203	497
T4 (ms)	207	337
T5 (ms)	201	454
T6 (ms)	198	436
T7 (ms)	204	445
T8 (ms)	206	332
T9 (ms)	202	326
T10 (ms)	198	362
Média (ms)	201	387
Impacto (ms)	186	

Como seria de esperar, o impacto das recomendações *Jarvis* no desempenho da função é baixo, cerca de 186 ms, obtendo uma resposta rápida e cumprindo um dos objetivos, ter pouco impacto na performance do EMS. Para além deste facto, os testes foram desenvolvidos para o *worst case*

scenario, sendo por isso, previsto que um utilizador comum não obtenha sequer este impacto, mas sim um inferior, pois normalmente os seus desportos favoritos têm eventos suficientes para preencher os valores do módulo.

5.3.3.2 All Markets

Seguidamente, foi analisado o tempo de resposta do módulo *All Markets*. Neste caso, os testes desenvolvidos foram semelhantes aos do *Popular Bets*, sendo que também foram realizados 20 testes (no anexo C.2) com e sem as recomendações ativas.

Tabela 5.17: Resultados do Impacto do *Jarvis* no módulo *All Markets*

	All Markets	
	sem *Jarvis*	com *Jarvis*
T1 (ms)	127	133
T2 (ms)	126	68
T3 (ms)	73	128
T4 (ms)	69	124
T5 (ms)	123	133
T6 (ms)	70	182
T7 (ms)	123	191
T8 (ms)	66	128
T9 (ms)	67	202
T10 (ms)	66	133
Média (ms)	93	140
Impacto (ms)	47	

Como é visível pela análise dos resultados obtidos na tabela 5.17, o aumento do tempo de resposta devido às recomendações é ainda mais reduzido, visto que aqui, a lógica inerente ao módulo é muito reduzida e apenas foi adicionado um pedido de dados aos serviços criados, sendo que dessa forma, o impacto medido é de aproximadamente 47 ms.

5.3.3.3 AZ

Para testar o módulo *AZ*, foram igualmente realizadas 20 medições (no anexo C.3) para avaliar a diferença no tempo de resposta do módulo com ou sem as recomendações.

Tabela 5.18: Resultados do Impacto do *Jarvis* no módulo *AZ*

	AZ	
	sem *Jarvis*	com *Jarvis*
T1 (ms)	292	591
T2 (ms)	298	641
T3 (ms)	283	607
T4 (ms)	335	606
T5 (ms)	254	654
T6 (ms)	271	638
T7 (ms)	270	677
T8 (ms)	289	679
T9 (ms)	268	662
T10 (ms)	288	620
Média (ms)	292	628
Impacto (ms)	336	

Analisando a tabela 5.18, é possível concluir que neste módulo houve um impacto de 336 ms, o que é um valor baixo mas assinalável tendo em conta os valores anteriores. Este módulo foi o que sofreu uma maior diferença de tempo de resposta, no entanto este valor deve-se ao facto de, ao contrário dos outros casos, este não possuir um serviço em *Backend* que envie os dados, mas sim em *Frontend*, o que irá implicar uma maior diferença no tempo de resposta. Por tudo isto, o impacto que o *Jarvis* tem neste módulo é de aproximadamente 336 ms, visto necessitar de pedidos adicionais no FE.

5.3.3.4 Serendipity

Posteriormente foi analisado o módulo *Serendipity*. Este não existia anteriormente e por isso, torna-se impossível avaliar qual o impacto causado com a sua implementação. Apesar disso, foi analisado qual o tempo que todo o módulo demorava na obtenção dos dados para este, para isso também foram realizados 20 testes (no anexo C.4).

Tabela 5.19: Resultados do Impacto do *Jarvis* no módulo *Serendipity*

Tempos (ms)	Serendipity
T1 (ms)	205
T2 (ms)	207
T3 (ms)	203
T4 (ms)	199
T5 (ms)	276
T6 (ms)	213
T7 (ms)	209
T8 (ms)	244
T9 (ms)	200
T10 (ms)	203
Média (ms)	216

Depois de analisar os resultados apresentados na tabela 5.19, é possível concluir que o tempo de carregamento dos dados é bastante baixo, cerca de 0,2 s, não forçando o EMS a um carregamento mais lento.

5.3.3.5 Top Recommendations

Depois de analisados todos os módulos, foi necessário investigar qual seria o tempo de resposta do próprio serviço criado para a comunicação entre o *Frontend* e o motor de recomendação *Jarvis*. Tal como anteriormente foram realizados 20 medições (no anexo C.5).

Tabela 5.20: Resultados do tempo de resposta do serviço *Top Recommendations*

Tempos (ms)	Top Recommendations
T1 (ms)	6
T2 (ms)	2
T3 (ms)	3
T4 (ms)	5
T5 (ms)	2
T6 (ms)	11
T7 (ms)	2
T8 (ms)	5
T9 (ms)	10
T10 (ms)	2
Média (ms)	3

Analisando a tabela 5.20, é possível concluir que o tempo de resposta deste serviço de dados é muitíssimo baixo, cerca de 3 ms. Este foi desenvolvido de forma a que as recomendações tivessem

um impacto mínimo na utilização do EMS e esta média de valores obtidos é a confirmação de que este é um serviço rápido e com baixo impacto na performance da aplicação.

5.4 Resumo

Depois de concluídos todos os testes, é possível verificar que no que toca à geração das recomendações, o tempo despendido nas tarefas é baixo, mas no que toca ao cálculo das semelhanças a performance não é ótima sendo mesmo o *bottleneck* de todo o processo, sendo que, apesar da enorme melhoria face à performance inicial, é um dos pontos a melhorar como trabalho futuro. Quanto aos módulos e serviços, o tempo de resposta é muito baixo e isso veio confirmar que a escolha do cálculo das recomendações uma vez por dia foi muito positiva, visto que proporciona uma melhor experiência de utilização, porque não muda as recomendações durante a utilização do apostador e, para além disso, diminui bastante o impacto do *Jarvis* no tempo de resposta do EMS. Por fim, depois de finalizados os testes é possível afirmar que os objetivos traçados em relação à eficiência e rapidez das recomendações foram devidamente cumpridos.

Capítulo 6

Conclusões

Os sistemas de recomendação têm como principal objetivo, fornecer aos *websites* ferramentas que permitam personalizar a informação disponibilizada aos utilizadores, possibilitando assim uma melhor experiência de utilização. Apesar de estes serem cada vez mais comuns, esta é ainda uma área que se encontra em expansão, em particular na vertente das apostas desportivas ainda pouco explorada. Neste âmbito, apresentou-se na presente dissertação um estudo sobre motores de recomendação, o estudo e desenvolvimento de uma solução de recomendação adequada ao ambiente *Betfair* e um estudo de performance sobre a solução desenvolvida.

Depois de compreendido qual o motivo e o problema que levou à realização desta dissertação, foi possível delinear os objetivos a alcançar, realizar um estudo acerca das abordagens existentes nos sistemas de recomendação, bem como os algoritmos de *Machine Learning* mais indicados para estes motores, das quais se podem destacar o *Naive Bayes* e o *K-Nearest Neighbours*. O estudo do estado da arte permitiu obter uma visão geral sobre os métodos já existentes acerca desta área, sobre algumas empresas que já utilizam estes sistemas, e sobre o enorme impacto que estes têm na experiência de utilização dos seus *websites*.

Relativamente à ferramenta desenvolvida, inicialmente foi realizado um estudo sobre qual tecnologia utilizar, efetuando uma comparação entre elas e depois concluir qual seria a mais indicada. Depois de tudo definido foi desenvolvida a ferramenta *Jarvis*, que consiste num motor de recomendação, que sugere ao utilizador quais os eventos mais indicados a este, baseando-se no seu histórico e também no de utilizadores semelhantes. Para todo este procedimento ser possível, foi necessária a integração de uma base de dados *Redis*, que ia registando alguns aspetos da navegação dos utilizadores. O *Jarvis*, recorre à base de dados para calcular recomendações, tratando os dados de forma a conseguir obter quais os desportos e mercados favoritos dos utilizadores. Finalizado o motor de recomendação, foi possível integrá-lo no EMS da *Betfair*, e desenvolver uma experiência melhorada e focada no utilizador, oferecendo mais destaque aos eventos favoritos deste.

Os testes efetuados à solução desenvolvida podem ser agrupados em testes de funcionalidade e de performance. No primeiro conjunto, foram desenvolvidos testes unitários (que verificam a funcionalidade de cada bloco de código), testes exploratórios (que certificam a funcionalidade ao navegar pela aplicação e utilizando uma base de dados com tendência conhecida, verifica que

os eventos sugeridos são correspondentes) e testes de validação dos processos (que certificam a funcionalidade de cada processo que integra a geração das recomendações). No segundo conjunto, foram realizados testes de performance (que medem o impacto do *Jarvis* nos tempos de resposta do EMS, seja nos pedidos como nos serviços desenvolvidos).

6.1 Contribuições

As contribuições do trabalho associado a esta dissertação são as seguintes:

- **Desenvolvimento de um motor de recomendação focado no utilizador.** Foi implementado um motor de recomendação num site de apostas desportivas que calcula as recomendações para cada utilizador e regista-as numa base de dados.
- **Integração de uma Base de Dados com o EMS.** A base de dados *Redis* foi integrada com o EMS para registo da atividade dos utilizadores.
- **Alteração do aspeto visual e do conteúdo do EMS.** Foram alterados os pedidos aos serviços para obter os eventos recomendados para o utilizador e moldado o aspeto do EMS para que este se adapte às recomendações calculadas para o utilizador.
- **Estudo dos resultados obtifos.** Foi realizado um estudo a nível de funcionalidade e performance em que foram analisados os resultados dos procedimentos e o impacto do *Jarvis* no EMS.

6.2 Resultados

Foi desenvolvida uma solução *Jarvis* que regista a atividade dos utilizadores, que calcula recomendações para estes e que altera o aspeto visual e o próprio conteúdo do EMS da *Betfair*, conseguindo melhorar a experiência de utilização da aplicação e facilitando a procura de mercados desejados, visto que são destacados os desportos e mercados recomendados para o utilizador. Esta solução foi desenvolvida em *Node.js* e *AngularJs*, e utiliza como base de dados *noSQL* o *Redis*.

6.3 Trabalho Futuro

Como trabalho futuro é possível expandir o leque de categorias registadas na base de dados, possibilitando aumentar o número de módulos sujeitos a recomendação. Por exemplo, a expansão das categorias registadas possibilitaria alterar os eventos numa página de desporto, ordenando-os consoante as recomendações do utilizador. Este aumento de categorias, apesar de possibilitar o aumento de módulos personalizados, poderia no entanto aumentar o impacto e piorar a performance da ferramenta. Para além disso, seria interessante alterar o peso do histórico de cada utilizador com o passar do tempo, ou seja, o valor atribuído a cada navegação seria dado por uma função a

tender para 0, tornando possível diminuir o peso de cada atividade ao longo do tempo em vez de ser removido do histórico ao final de 180 dias. Como contribuições futuras, seria ainda interessante melhorar a performance do algoritmo, sobretudo no método de cálculo das recomendações. Outro ponto para desenvolvimento futuro, seria a criação de uma página de estatísticas em que o utilizador, por exemplo, poderia consultar quais os desportos favoritos, quais os eventos mais apostados, entre outros.

Anexo A

Aspeto Visual do EMS

A.1 EMS sem *Jarvis*

A.1.1 EMS para *Smartphones* sem *Jarvis*

A imagem apresentada de seguida corresponde ao ambiente gráfico do EMS para *smartphones*.

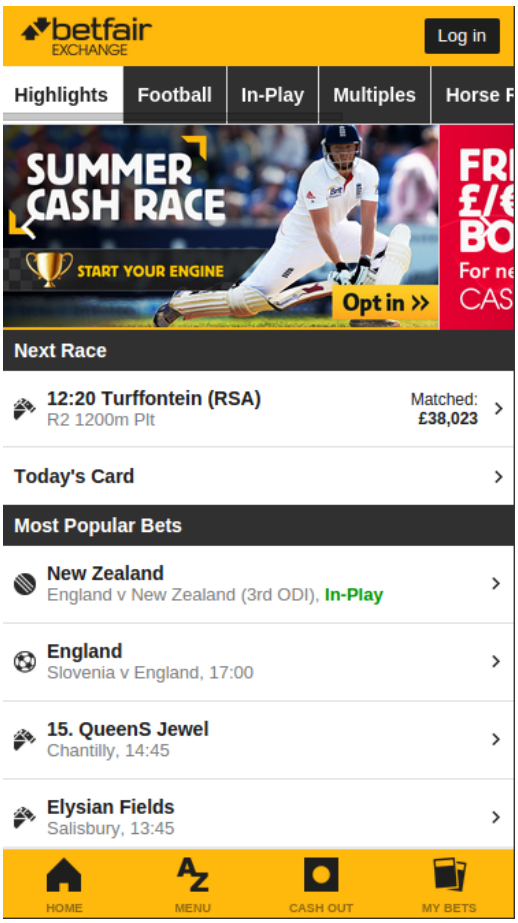

Figura A.1: Representação do EMS para *Smartphone*

A *homepage* do EMS para *smartphones* pode ser dividida nos seguintes módulos:

1. Ribbon
2. Next Race
3. Most Popular Bets

A.1.2 EMS para *Tablets* sem *Jarvis*

A imagem apresentada de seguida corresponde ao ambiente gráfico do EMS para *tablets*.

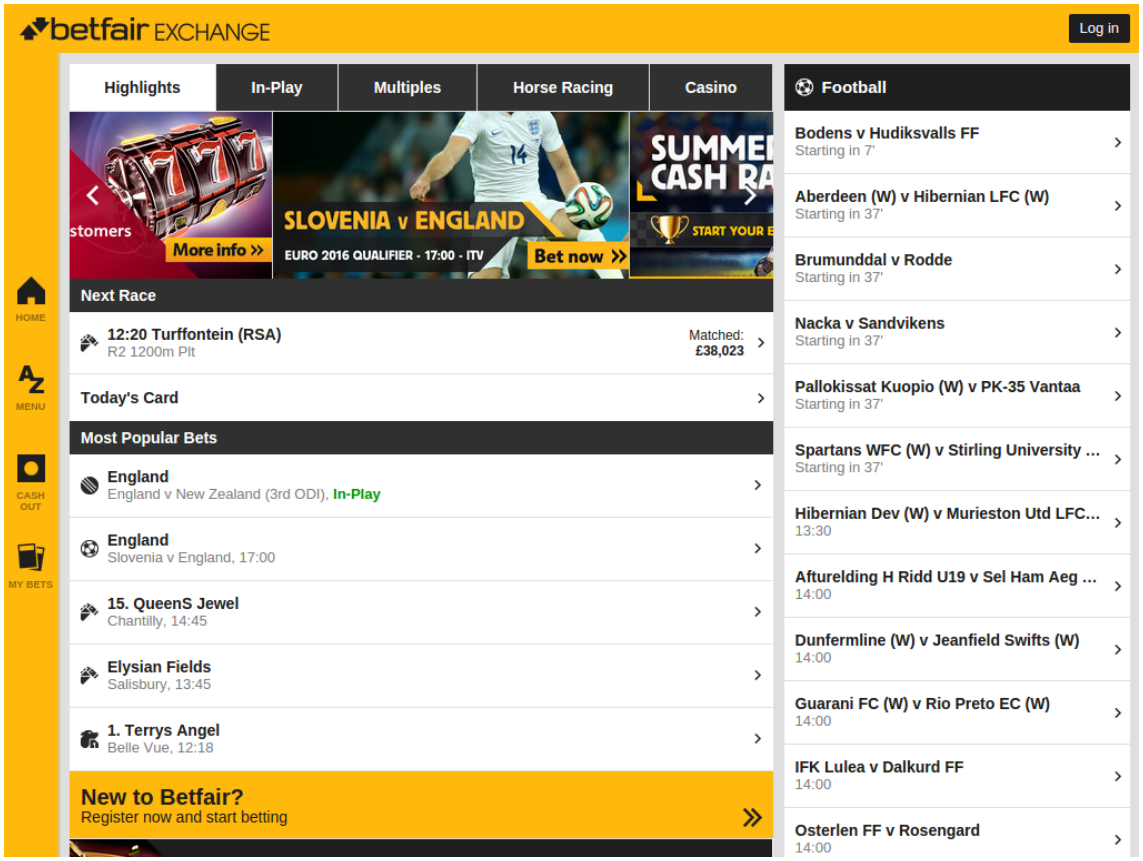

Figura A.2: Representação do EMS para *Tablet*

A *homepage* do EMS para *tablets* pode ser dividida nos seguintes módulos:

1. Ribbon
2. Next Race
3. Side Menu
4. Most Popular Bets

A.2 EMS com *Jarvis*

A.2.1 EMS para *Smartphones* com *Jarvis*

A imagem apresentada de seguida corresponde ao ambiente gráfico do EMS para *smartphones*.

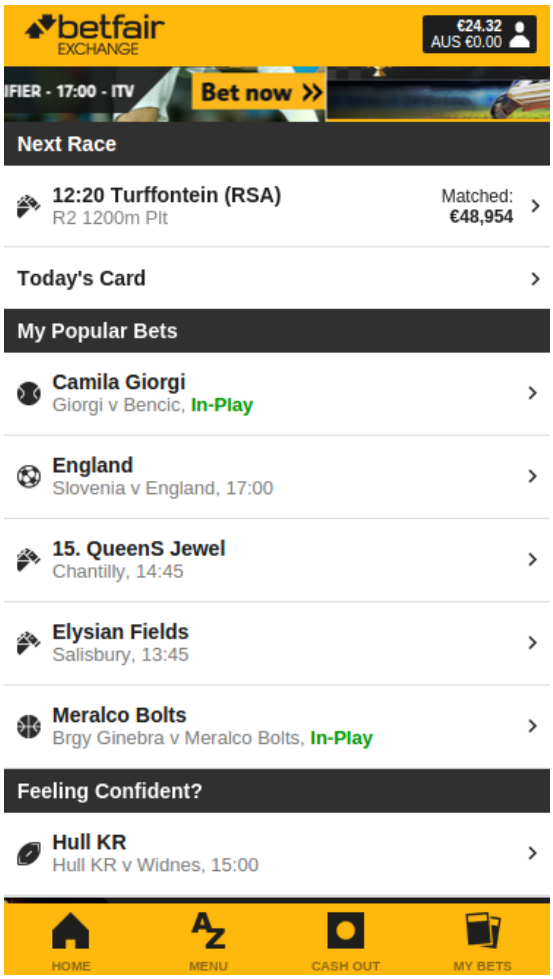

Figura A.3: Representação do EMS para *Smartphone* com *Jarvis*

A *homepage* do EMS para *smartphones* pode ser dividida nos seguintes módulos:

1. Ribbon
2. Next Race
3. Most Popular Bets
4. Serendipity

A.2.2 EMS para *Tablets* sem *Jarvis*

A imagem apresentada de seguida corresponde ao ambiente gráfico do EMS para *tablets*.

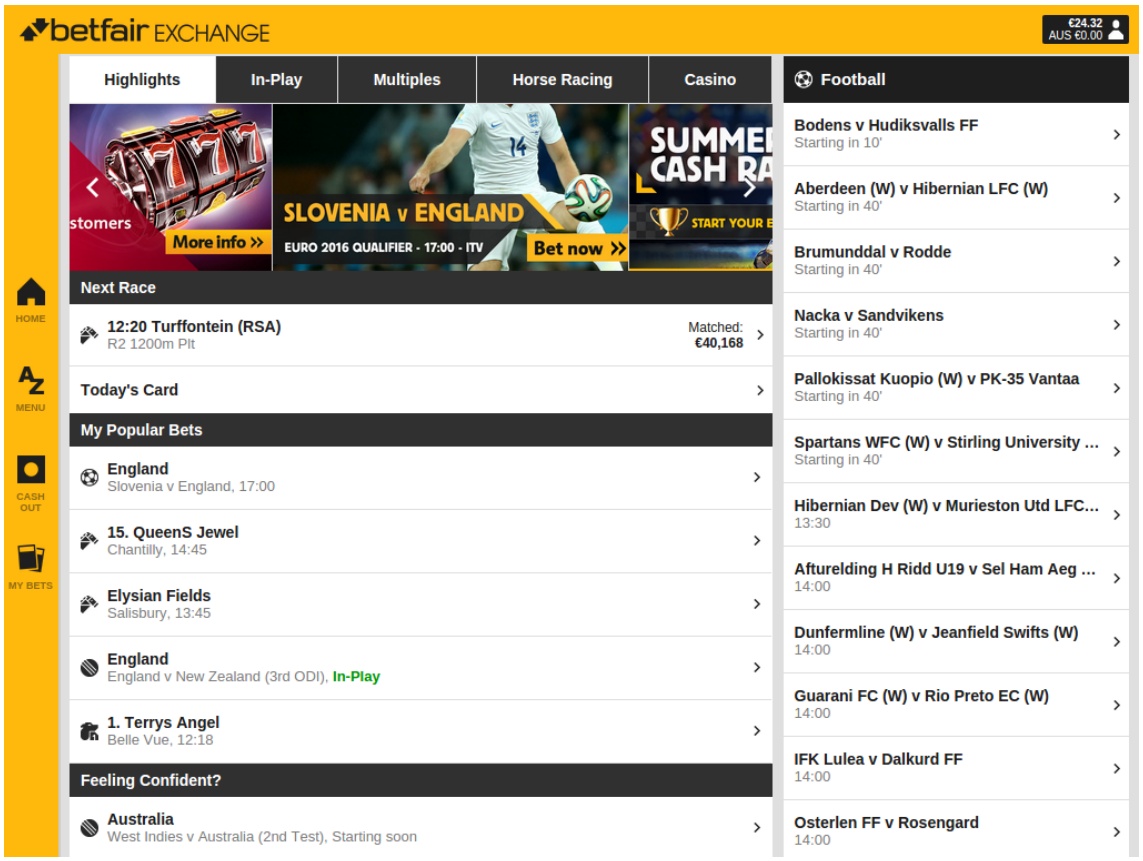

Figura A.4: Representação do EMS para *Tablet* com *Jarvis*

A *homepage* do EMS para *tablets* pode ser dividida nos seguintes módulos:

1. Ribbon
2. Next Race
3. Side Menu
4. Most Popular Bets
5. Serendipity

Anexo B

Valores do Histórico de Utilizadores

B.1 Contagens e Probabilidades de Utilizadores

Tabela B.1: Contagens e Probabilidades do Utilizador u10

Ids de Desporto e Mercado	Contagens	Probabilidades
7524	200	0.50
7511	100	0.25
6	60	0.15
7	40	0.10
TO_SCORE	50	0.50
OVER_UNDER_25	30	0.30
OVER_UNDER_05	20	0.20

Tabela B.2: Contagens e Probabilidades do Utilizador u5

Ids de Desporto e Mercado	Contagens	Probabilidades
7524	100	0.50
7511	50	0.25
7	30	0.15
6	20	0.10
TO_SCORE	50	0.50
OVER_UNDER_25	30	0.30
OVER_UNDER_05	20	0.20

Tabela B.3: Contagens e Probabilidades do Utilizador u20

Ids de Desporto e Mercado	Contagens	Probabilidades
7524	200	0.50
7511	100	0.25
7	60	0.15
7522	40	0.10
TO_SCORE	50	0.50
OVER_UNDER_25	30	0.30
OVER_UNDER_05	20	0.20

Tabela B.4: Contagens e Probabilidades do Utilizador u15

Ids de Desporto e Mercado	Contagens	Probabilidades
7524	100	0.50
7511	50	0.25
6	30	0.15
1	20	0.10
TO_SCORE	50	0.50
OVER_UNDER_25	30	0.30
MATCH_ODDS	20	0.20

Anexo C

Testes de Performance

C.1 Popular Bets

Tabela C.1: Resultados do Impacto do *Jarvis* no módulo *Popular Bets*

	Popular Bets	
	sem *Jarvis*	com *Jarvis*
T1 (s)	203	507
T2 (s)	208	392
T3 (s)	203	497
T4 (s)	207	337
T5 (s)	201	454
T6 (s)	198	436
T7 (s)	204	445
T8 (s)	206	332
T9 (s)	202	326
T10 (s)	198	362
T11 (s)	200	329
T12 (s)	196	323
T13 (s)	208	333
T14 (s)	202	322
T15 (s)	197	327
T16 (s)	197	440
T17 (s)	198	455
T18 (s)	202	384
T19 (s)	194	386
T20 (s)	200	486
Média (s)	201	387
Impacto (s)	186	

C.2 All Markets

Tabela C.2: Resultados do Impacto do *Jarvis* no módulo *All Markets*

	All Markets	
	sem *Jarvis*	com *Jarvis*
T1 (ms)	127	133
T2 (ms)	126	68
T3 (ms)	73	128
T4 (ms)	69	124
T5 (ms)	123	133
T6 (ms)	70	182
T7 (ms)	123	191
T8 (ms)	66	128
T9 (ms)	67	202
T10 (ms)	66	133
T11 (ms)	67	93
T12 (ms)	138	69
T13 (ms)	67	67
T14 (ms)	68	179
T15 (ms)	71	179
T16 (ms)	145	178
T17 (ms)	67	187
T18 (ms)	126	124
T19 (ms)	127	123
T20 (ms)	125	179
Média (ms)	93	140
Impacto (ms)	47	

C.3 AZ

Tabela C.3: Resultados do Impacto do *Jarvis* no módulo *AZ*

	AZ	
	sem *Jarvis*	com *Jarvis*
T1 (ms)	292	591
T2 (ms)	298	641
T3 (ms)	283	607
T4 (ms)	335	606
T5 (ms)	254	654
T6 (ms)	271	638
T7 (ms)	270	677
T8 (ms)	289	679
T9 (ms)	268	662
T10 (ms)	288	620
T11 (ms)	291	623
T12 (ms)	329	609
T13 (ms)	276	595
T14 (ms)	265	576
T15 (ms)	323	595
T16 (ms)	260	656
T17 (ms)	320	666
T18 (ms)	357	595
T19 (ms)	317	654
T20 (ms)	270	630
Média (ms)	292	628
Impacto (ms)	336	

C.4 Serendipity

Tabela C.4: Resultados do Impacto do *Jarvis* no módulo *Serendipity*

Tempos (ms)	Serendipity
T1 (ms)	205
T2 (ms)	207
T3 (ms)	203
T4 (ms)	199
T5 (ms)	276
T6 (ms)	213
T7 (ms)	209
T8 (ms)	244
T9 (ms)	200
T10 (ms)	203
T11 (ms)	204
T12 (ms)	206
T13 (ms)	209
T14 (ms)	196
T15 (ms)	208
T16 (ms)	209
T17 (ms)	208
T18 (ms)	206
T19 (ms)	254
T20 (ms)	256
Média (ms)	216

C.5 Top Recommendations

Tabela C.5: Resultados do tempo de resposta do serviço *Top Recommendations*

Tempos (ms)	Top Recommendations
T1 (ms)	6
T2 (ms)	2
T3 (ms)	3
T4 (ms)	5
T5 (ms)	2
T6 (ms)	11
T7 (ms)	2
T8 (ms)	5
T9 (ms)	10
T10 (ms)	2
T11 (ms)	2
T12 (ms)	8
T13 (ms)	3
T14 (ms)	2
T15 (ms)	2
T16 (ms)	2
T17 (ms)	7
T18 (ms)	1
T19 (ms)	1
T20 (ms)	1
Média (ms)	3

Referências

[1] Anand Rajaraman e Jeffrey D Ullman. *Mining of Massive Datasets*, volume 67. 2011. URL: http://ebooks.cambridge.org/ref/id/CBO9781139058452, doi:10.1017/CBO9781139058452.

[2] Marko Sarstedt e Erik Mooi. *A Concise Guide to Market Research*. Springer Texts in Business and Economics. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. URL: http://link.springer.com/10.1007/978-3-642-53965-7, doi:10.1007/978-3-642-53965-7.

[3] G. Castellano, a.M. Fanelli, e M.a. Torsello. NEWER: A system for NEuro-fuzzy WEb Recommendation. *Applied Soft Computing*, 11(1):793–806, Janeiro 2011. URL: http://linkinghub.elsevier.com/retrieve/pii/S1568494610000049, doi:10.1016/j.asoc.2009.12.040.

[4] Betfair. Betfair, 2014. URL: http://www.betfair.com/exchange.

[5] Utpala Niranjan, Kanduri Srividya, V. V. Krishna, e V. Khanaa. Developing a dynamic web recommendation system based on incremental data mining. *2011 3rd Int. Conf. Electron. Comput. Technol.*, páginas 247–252, Abril 2011. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5941747, doi:10.1109/ICECTECH.2011.5941747.

[6] Mukund Deshpande e George Karypis. Recommendation Algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004. arXiv:34, doi:10.1145/963770.963776.

[7] Amazon. Amazon.co.uk: Low Prices in Electronics, Books, Sports Equipment & more. URL: http://www.amazon.co.uk/ [último acesso em 2015-01-16].

[8] Netflix. Netflix - Watch TV Shows Online, Watch Movies Online. URL: https://www.netflix.com/global [último acesso em 2015-01-16].

[9] Pandora. Pandora personalized radio. URL: http://www.pandora.com [último acesso em 2015-01-16].

[10] Lastfm. Last.fm - Listen to free music and watch videos with the largest music catalogue online. URL: http://www.last.fm/ [último acesso em 2015-01-16].

[11] Gediminas Adomavicius e Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. arXiv:3, doi:10.1109/TKDE.2005.99.

[12] Badrul Sarwar, George Karypis, Joseph Konstan, e J Riedl. Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th ...*, 1:285–295, 2001. URL: http://dl.acm.org/citation.cfm?id=372071, arXiv:119, doi:10.1145/371920.372071.

[13] Michael D. Ekstrand, John T. Riedl, e Joseph A. Konstan. Collaborative Filtering Recommender Systems. *Found. Trends® Human–Computer Interact.*, 4(2):81–173, 2011. doi:10.1561/1100000009.

[14] P Resnick e Hr Varian. Recommender systems. *Commun. ACM*, páginas 1–21, 1997. URL: http://dl.acm.org/citation.cfm?id=245121, doi:10.1162/153244302760200641.

[15] NJ Nilsson. Introduction to machine learning-an early draft of a proposed textbook. Department of Computer Science. 1996. URL: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:INTRODUCTION+TO+MACHINE+LEARNING+AN+EARLY+DRAFT+OF+A+PROPOSED+TEXTBOOK+Department+of+Computer+Science#0http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Introduction+to+machine+learning-an+early+draft+of+a+proposed+textbook.+Department+of+Computer+Science#0.

[16] Thomas Dietterich, Christopher Bishop, David Heckerman, Michael Jordan, e Michael Kearns. *Introduction to Machine Learning Second Edition*. 2010.

[17] M.a. Friedl e C.E. Brodley. Decision tree classification of land cover from remotely sensed data. *Remote Sensing of Environment*, 61:399–409, 1997. doi:10.1016/S0034-4257(97)00049-7.

[18] Dell. Naive Bayes classifier, 2006. URL: http://www.statsoft.com/textbook/naive-bayes-classifier [último acesso em 2015-05-18].

[19] H Barlow. Unsupervised learning. *Neural computation*, páginas 1–32, 1989. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6796603.

[20] AK Jain, MN Murty, e PJ Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3), 1999. URL: http://dl.acm.org/citation.cfm?id=331504.

[21] Christopher D. Manning, Prabhakar Raghavan, e Hinrich Schütze. *An Introduction to Information Retrieval*. Número c. Cambridge University Press, 2009.

[22] Fionn Murtagh e Pedro Contreras. Methods of Hierarchical Clustering. (2):1–21, Abril 2011. URL: http://arxiv.org/abs/1105.0121v1, arXiv:1105.0121.

[23] Sami Ayramo e Tommi Karkkainen. *Introduction to partitioning-based clustering methods with a robust example*, volume 35. University of Jyvaskyla, 2006.

[24] Raymond T. Ng e Jiawei Han. Efficient and Effective Clustering Methods for Spatial Data Mining. páginas 1–25, 1994.

[25] Daniela Şchiopu. Applying TwoStep cluster analysis for identifuing bank customers' profile. *Buletinul*, LXII(3):66–75, 2010. URL: http://upg-bulletin-se.ro/archive/2010-3/7.Schiopu.pdf.

[26] Bing Liu, Wynne Hsu, e Yiming Ma. Integrating classification and association rule mining. *KDD-98 Proceedings*, página 6, 1998. URL: http://www.aaai.org/Papers/KDD/1998/KDD98-012.pdf.

[27] Rakesh Agrawal, Tomasz Imieliński, e Arun Swami. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*, (May):207–216, 1993. URL: http://portal.acm.org/citation.cfm?doid=170035.170072, doi:10.1145/170035.170072.

[28] Jochen Hipp, Ulrich Guntzer, e Gholamreza Nakhaeizadeh. Algorithms for Association Rule Mining - A General Survey and Comparison. *SIGKDD Explorations*, 2(1):58–64.

[29] Petr Berka e Jan Rauch. Machine Learning and Association Rules. *rocq.inria.fr*, (2003). URL: https://www.rocq.inria.fr/axis/COMPSTAT2010/TU_Berka-Rauch_paper.pdf.

[30] Sotiris Kotsiantis e Dimitris Kanellopoulos. Association Rules Mining : A Recent Overview Basic Concepts & Basic Association Rules Algorithms. 32(1):71–82, 2006.

[31] Rakesh Agrawal e Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. *VLDB '94 Proc. 20th Int. Conf. Very Large Data Bases*, páginas 487–499, 1994. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.7506, doi:10.1.1.40.7506.

[32] Erik Strumbelj e Marko Robnik Sikonja. Learning Betting Tips from Users ' Bet Selections. páginas 678–688, 2009.

[33] D Miljkovic e L Gajic. The use of data mining for basketball matches outcomes prediction. *Intelligent Systems and . . .*, páginas 309–312, 2010. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5647440.

[34] AngularJS. AngularJS — Superheroic JavaScript MVW Framework. URL: https://angularjs.org/ [último acesso em 2015-05-20].

[35] NodeJS. Node.js. URL: https://nodejs.org/ [último acesso em 2015-05-20].

[36] Karma. Karma - Spectacular Test Runner for Javascript. URL: http://karma-runner.github.io/0.12/index.html [último acesso em 2015-05-20].

[37] Jasmine. Jasmine: Behavior-Driven JavaScript. URL: http://jasmine.github.io/ [último acesso em 2015-05-20].

[38] Chai. Chai. URL: http://chaijs.com/ [último acesso em 2015-05-20].

[39] Sinon.JS. Sinon.JS - Documentation. URL: http://sinonjs.org/ [último acesso em 2015-05-20].

[40] Redis. Redis. URL: http://redis.io/ [último acesso em 2015-05-20].

[41] JSLint. JSLint:The JavaScript Code Quality Tool. URL: http://www.jslint.com/ [último acesso em 2015-06-12].

[42] PredictionIO. PredictionIO Open Source Machine Learning Server. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5941747 [último acesso em 2015-01-21].

[43] Raccoon. Recommendation Raccoon, 2015. URL: https://github.com/guymorita/recommendationRaccoon [último acesso em 2015-05-20].

[44] HBase. HBase – Apache HBase™ Home. URL: http://hbase.apache.org/ [último acesso em 2015-05-20].

[45] IEEE Standards Board. *An American National Standard IEEE Standard for Software Unit Testing*. 1993.